Institut d'études politiques de Paris

ECOLE DOCTORALE DE SCIENCES PO

Programme doctoral de droit

Le Centre de Recherche de l'École de Droit

Doctorat en Droit

Story of a Legal Codex(t)

Writing Law in Code

Megan Ma

Thesis supervised by Horatia MUIR WATT, Full Professor and Co-Director of the Global Governance Studies program

defended on 10th December 2021

Jury:

Mireille HILDEBRANDT, Professor, Radboud University

Daniel W. LINNA Jr., Senior Lecturer, Northwestern Law School (reviewer)

Harry SURDEN, Professor of Law, University of Colorado Law School (reviewer)

David WINICKOFF, Principal Analyst, OECD

Story of a Legal Codex(t):
Writing Law in Code

Megan Ma
Sciences Po Law School

Supervisors: Horatia Muir Watt (Director), David Winickoff (Minor)
Jury: Mireille Hildebrandt, Daniel W. Linna, Harry Surden

This dissertation is submitted for the degree of
PhD in Law

Foreword and Acknowledgments

This thesis is inspired by a childhood pastime of mine: speaking in a language of my own creation. I was curious about why words corresponded with specific meanings, questioning whether I could stretch words beyond their ascribed understanding. Though what I was doing was, in fact, creating my own dialect –a mutant variant of the English language – the thesis is, in part, drawn from my fascination towards the unique linguistic vessel of natural language. How we are able to construct and reconstruct with natural language, its illimitable malleability, continues to be a source and driver of my scholarly pursuits. And so, as we dive deeper and venture further into the next era of text, my hope is that we may be able to better understand how code is able to write and create the stories of tomorrow.

This dissertation would not be possible without the immense support of my community. I am indebted to so many, notably:

(a) Horatia Muir Watt who is a well of knowledge, source of light, a mentor and role model that one could only dream about having.

(b) David Winickoff for ensuring that my research was always well-grounded, clear, and structured. His feedback has always pushed me to new heights.

(c) Adam Nicholas for dedicating hours of his time, sharing his perspective and providing rich feedback on my work.

(d) Dazza Greenwood and Bryan Wilson for their initial "hack" session on the use of programming languages in contracts that steered my thesis towards its current direction.

(e) Roland Vogl and Mike Genesereth for their continued encouragement and motivation to pioneer work in this area of research.

Lastly, I dedicate this thesis to my family. Without them, I would not dare to dream and, certainly, could not imagine being here today able to complete this thesis. To them, I owe everything.

PROLOG(UE)

How is the law measured? This is perhaps a leading question. For long, it appeared that the law cannot be measured. While there are standards and processes, the law was not regarded as quantifiable. Only in the advent of recent technological advancements have there been considerations for metrics.[1] The range of technology used in the field of law has been rather vast and variable. Yet, they have all pointed towards increasing the capacity to measure the law. These arguments speak towards the legal field's inherent protectionism, enabled by knowledge possessed by a privileged "class of individuals."[2] This has erected and perpetuated barriers to access owed to information asymmetries.[3] Consequently, the rise in 'legal analytics,' or a metrics for law, has stemmed from an access to justice perspective. The assumption is that in making the law more quantifiable, knowledge that has been historically opaque and inaccessible outside of the legal community may be revealed.

In unpacking the law, recurring arguments around the integration of computational technology in legal practice have centered on the incomprehensibility and complexity of the legal language. Proposed solutions include automating legal documents or using machine learning technology and/or neural networks to demystify patterns of court behavior. These technologies have all brought to light new quantitative methods of evaluation. Nevertheless, it appears that they pivot around a deeper linguistic problem. Beneath the fervor of technological enthusiasm is the desire to better understand the language of legal processes.

Alternatively, it may be argued that the law has always been measurable. Words, through linguistic devices, have shaped legal meaning. In effect, the law conceivably has been measured by its words. Evidently, the use of "the law" is rather vague. It ineptly personifies the discipline and removes its actors, history, and institutions. It may be clarified here that reference to the law, for the purpose of this dissertation, is reference to written legal text. While there are other mediums 'the law' uses to communicate, written text is frequently considered the primary site for legal interpretation. In fact,

---

[1] Consider, for example, recent discussions around quality in legal work. *See* David Cunningham, "Metrics of the NewLaw Model," *Legal Evolution* (Oct. 18, 2020) https://www.legalevolution.org/2020/10/metrics-of-the-newlaw-model-206/. *See* also John Armour and Mari Sako, *AI-enabled business models in legal services: from traditional law firms to next generation law companies*?, 7 JOURNAL OF PROFESSIONS AND ORGANIZATIONS 27-46 (2020).

[2] Joshua Browder, "Law as Code: A Legal System Shaped by Software, *Future* (Jun. 15, 2021) https://future.a16z.com/law-as-code/.

[3] Daniel W. Linna Jr., *The Future of Law and Computational Technologies: Two Sides of the Same Coin*, MIT COMPUTATIONAL LAW REPORT Release 1.0 (2019) available at: https://law.mit.edu/pub/thefutureoflawandcomputationaltechnologies/release/2.

"law *exists as* text."[4] I further this line of thinking by questioning the vehicle of natural language. That is, natural language has been the key vessel through which the law has manifested itself. Does the law then depend on natural language to do its work? Importantly, is the language sufficient at housing legal norms?

This dissertation, therefore, seeks to tell a narrative. Broadly, it chronicles the story of law's intimate relationship with language. But more specifically, the thesis details the law's recent encounter with the digital. When law met technology, its relationship with language changed, invoking skepticism around its fitness for the conveyance of legal concepts. With the introduction of an innovative player – code – the law had perceivably found its new linguistic match. As a result, code was tested for its ability to perform and accommodate for the law's demands. Ultimately, confronted by natural language and code, the law is asked whether code can be its language.

The dissertation aims to put forth the following thematic discussions. First, the legal language is a social phenomenon, whereby form and substance are inseparable. The distinct characteristics of the language are inherent to its formulation. This reaffirms the notion that law is a "relational construct"[5] that belongs to a broader discursive formation. It is a network understanding of both the internal ordering and relationship to other discourses. In other words, the legal language mediates between societal expectations and the formal procedure that enacts constraints and rights to parties involved.[6] Further to this thought, the legal language is necessarily rich because it is a "historical artifact."[7] The complexity of its concepts is woven from its contextual environment and is the result of natural evolution; in effect, "generat[ing] continuity and durability."[8] Accordingly, legal concepts cannot simply be divorced from its linguistic encasing.

This then leads to my next argument. There is a sharp distinction between clarity and simplicity. That is, simplification does not necessarily lead to clarification. They are false cognates and should not be treated as equivalents. It shall be demonstrated that attempts at simplifying the language not only are futile, but also inadvertently reduce legal complexity and muddy the significance of tradition

---

[4] Mireille Hildebrandt, "Intricate entanglements of law and technology," in *Smart Technologies and the End(s) of Law: Novel Entanglements of Law and Technology* 161 (2015).

[5] *Id.* at 172.

[6] *Id.* at 173-174.

[7] *Id.*

[8] *Id.* at 177.

in law. Simplification fosters the effect that legal norms exist independently of their environment, creating the illusion that the language is intended only for communication. Furthermore, the process of simplification alludes to the gap between the language and the embedded norm. Through simplification, the belief is that this gap can and should be closed.

Legal fictions, on the other hand, are a linguistic phenomenon that represents the points at which legal language stops communicating.[9] Legal fictions are fossilized metaphors, that, though are consciously counterfactual propositions, remain fundamental to the language. Importantly, legal fictions are both historically contingent and assertions of 'fact' that depend solely on the relations and powers effectuated by specific legal realities.[10] This suggests that clarifying legal language is not merely a matter of simplifying its communicative function. Instead, clarification involves epistemological deconstruction. I hope to illustrate that conflating simplification with clarification not only flattens the law, but also, fuels issues of translation in the context of 'code-ification.'

Third, the characteristics of legal language are, in fact, the characteristics of natural language. This is perhaps trite, but I consider that, to properly gauge the relationship between law and language, what must first be understood is the linguistic makeup of natural language itself. This allows for a deeper investigation into the processes involved in the construction of legal concepts. Linguistic theory, therefore, provides insight into how "interpretation becomes the hallmark of law."[11] Moreover, it reveals how language can intrinsically embody authority and be made objective and logical. Developing, then, an understanding of how natural language is built by its linguistic pillars – syntax, semantics, and pragmatics – the nuances of legal text are revealed. That is, how legal language formulates fact, creates reference and implicature, and upholds conscious falsities confronts both the boundaries and requirements to "sustain [the law's] identity."[12]

This thesis, then, traces the specific linguistic qualities that preserve and "root"[13] law in natural language. More importantly, I use these qualities to test against the competencies of computer code as legal language. The conclusions that may be drawn are paradoxical. On the one hand,

[9] Karen Petroski, *Legal fictions and the limits of legal language*, 9 INT. J. OF L. IN CONTEXT 485 (2013).

[10] *Id.* at 497.

[11] Hildebrandt, *supra* 4 at 177.

[12] *Id.* at 159.

[13] *Id.* at 174.

programming languages cannot draft legal text, if they are conceived solely for their logical and functional traits. On the other, in reconceptualizing code as a linguistic medium, and thereby accounting for its aesthetic dimension, code *is* perceivably a form of legal writing. Though these arguments appear to be rather theoretical, the implications are, in fact, significant.

As mentioned, the rapid technological advancements in computation have placed immense pressure on the legal system to change. Specifically, the law is regarded as 'trapped' in an antiquated and analog form; and that software is the answer. This claim is, of course, laced with technological solutionism.[14] Moreover, it falls in line with the aforementioned problems of simplification. While it is not my intention to suggest that software and computational technologies have no place in the legal realm, I consider a subtler argument. That is, for the furtherance of computational law, it cannot be done so from an architectural standpoint. Software code cannot simply conduct legal tasks. Conceiving code as application-based and task-oriented not only threatens to reconfigure law as logical reductions, but also has the potential to erase law's mode of existence.[15] Should law exist as text, code must, therefore, be analyzed at a linguistic level. Consequently, the tension to digitize requires the attention from scholars on how code, as *writing*, must find methods of reconciling its own practices and norms with existing legal norms. This dissertation is, thus, a contribution to the existing body of legal scholarship in two-fold: (1) to see code as interpretable; and (2) to introduce the hermeneutics of code to the legal space.

To tackle these discussions, the dissertation will unfold as follows. The remainder of the *Prolog(ue)* will form the background, situating the existing scholarly discussion. The dissertation will then transition into its first substantive chapter, *The Linguistic Affair*, revisiting the seminal conversations around law and language. The chapter will walk through various perspectives on the unique behaviors of legal language and reflect on the tensions surrounding interpretation. These include: (1) the difference between clarity and precision; (2) the paradox of form and substance; and (3) the myths of the fact-law distinction. Structurally, the chapter follows three key dimensions of the

[14] The definition is one described by Evgeny Morozov, "an endemic ideology that recasts complex social phenomena as neatly definable problems with definite, computable solutions, or as transparent and self-evident processes that can be easily optimize." *See* Evgeny Morozov, *To Save Everything, Click Here: The Folly of Technological Solutionism* (2013).

[15] To clarify, I am considering specifically Hildebrandt's definition that the law is relational, a co-dependence forming between information and communication infrastructures and modern positive law. *See* Hildebrandt, *supra* 4 at 172.

relationship between law and language: (1) the language of law; (2) law's language; and (3) law as language. The chapter culminates in an assessment of natural language as the vehicle for legal writing.

The next chapter, *Language Lego*, is a disciplinary bridge between linguistics and computer programming. It provides the grounds for linguistic analysis that moves beyond philosophy. More importantly, it hopes to debunk the misconceptions and misnomers around syntax and semantics in linguistics relative to computation. This chapter effectively provides the foundational tools for the remainder of the dissertation. The following chapter, *Case Studies on Translation*, is a three-part series that investigates the translation of law to code. Each case study analyzes how legal text has been transformed into code. The first case study explores computable contracts, while the third considers machine-readable legislation. The second case study stands apart from the other two. As opposed to analyzing translations of text to code, the second case study attempts to translate judicial decisions into code using a combined linguistic and statistical method.

The penultimate chapter, *Weaving the Code*, ties together observations from the case studies with the theoretical discussion. Perhaps as the crux of the dissertation, the chapter will introduce the problem with inference, then proceed with a thought experiment on code as the next legal language. More specifically, I draw attention towards potential methods of developing a legal semiotics. I advance the notion of legal codex(t): a simultaneous jeu de mots on computer code, conceptualizing code as text, and the term *codex*, signifying ancestry (ancestor) of text. Importantly, legal codex(t) is symbolic of the future of computational law for which I am hopeful to see. It is one that is sensitive to the histories and context inherent in legal norms. More importantly, legal codex(t) seeks to embody what natural language can do, capturing the linguistic and evolutionary nuances in the construction of meaning, while also counteracting where natural language has faltered. Finally, the dissertation will conclude with its *Epilog(ue)*. This chapter will further the ideas put forth in *Weaving the Code* to then acknowledge the emerging horizons of code as legal expression.

Prior to delving into the literature review, several 'terms of art' must be defined. These are: (1) context; (2) formal/formalize/formalism; (3) efficiency; and (4) code/ code-ification. To start, *context* is defined both in the broadest semiotics and linguistics sense of the term. That is, it refers to the knowledge, both tacit and explicit, that surrounds a particular text and is informative of its meaning. Second, I distinguish between the terms, *formal* and *formalize*. *Formal* is used interchangeably with logical and highly structured (as is found in programming languages). *Formalize*, though related, refers specifically to the act of standardizing and incorporating structure. *Formalism*, on the other

hand, is slightly more complex. I shift between theor(ies) of formalism and the state of being structured. As will be seen in the first case study, I engage in a play on words. The triad of *formal/formalize/formalism* will allude to the role of structure as it intersects across law, linguistics, and computation. Third, I frequently refer to the notion of *efficiency*. I define *efficiency* most consistently with the law and economics sense of the word, in particular, on the minimization of transaction costs and economic optimization of the legal system. Finally, *code* is used broadly with programming languages as well as the act of programming. *Code-ification* refers to the act of translating from law to code. Interestingly, it is a play on codification. As codification is the process involved with inscribing legal norms, code-ification is a commentary around code's competence to write the law. Having established these terms of art, this dissertation will now turn to the scholarly background in which it is seated.

A. STAGING

The digitization of society has raised the attention of scholars on the future. Whether the future of employment,[16] the future of healthcare, or the future of education, etc., the anticipation has mounted to a dualism of fear and excitement. The advent of AI, in particular, has struck a chord. But, in recent years, this chord has echoed so loudly that the fervor around the subject matter has led many to believe that AI is, in fact, "magical fairy dust."[17] Moreover, the literature has since become so vast that conversation on AI has been rendered nearly impenetrable, with experts readily deploying buzzwords that virtually have lost any meaning.[18]

Nevertheless, there is merit in reflecting on the narratives that have been constructed around AI and the lure of the machine. The remainder of this chapter seeks to survey the scholarly grounds on which AI has come to be understood and imagined; the stories that have been crafted about technology for humanity. Delving first into the mythology, the section then advances into the initial reactions and proposed responses to AI. As the intention of the dissertation is to unpack the notion

[16] Daniel Susskind, *A World Without Work* (2020). *See* also Daniel Susskind and Richard Susskind, *The Future of the Professions: How Technology will Transform the Work of Human Experts* (2015); and Alex Rosenblat, *Uberland: How Algorithms are Rewriting the Rules of Work* (2018).

[17] The suggestion of mentally replacing all mentions of "AI" in an article with the term "magical fairy dust." *See* Jeremy Hsu, "3 Easy Ways to Evaluate AI Claims," IEEE Spectrum (Aug. 23, 2019) https://spectrum.ieee.org/tech-talk/artificial-intelligence/machine-learning/learn-the-red-flags-of-overhyped-ai-claims.

[18] Consider the definition of blockchain and smart contracts. *See* for example, Adrianne Jeffries, "'Blockchain' is Meaningless," The Verge (Mar. 7, 2018) https://www.theverge.com/2018/3/7/17091766/blockchain-bitcoin-ethereum-cryptocurrency-meaning.

of computation and law, I consider uniquely the legal space and how AI has been discussed in relation to it.

The literature review will progress into questions of whether the law is computable and whether there is an inherent shift in its philosophy in light of technological integration. The section subsequently pivots, highlighting that existing literature regards the field of AI and law through a fundamentally macrosystemic lens and fails to account for a micro-level analysis. That is, in reconciling the computability of law with computational law, I argue that it is perhaps more important to consider beyond a wholesale regard of the field. Instead, a deeper analysis into the mechanics and language offer a more critical perspective. The section will then conclude by working through texts from the emerging discipline of legal analytics and informatics. This chapter is, in effect, one of stage-setting. Therefore, to better contextualize the analytical background, it is important to start from the beginning.

From Mythology to Technological Utopia

When asked to visualize AI in the mind's eye, what does one imagine? Adrienne Mayor argues that the first images of AI sparked in Greek mythology[19] with ideas and designs of "artificial life." She describes myths as thought experiments on entities that are "made, not born."[20] These entities – automatons, as she calls – were considered products of *biotechne*, life through craft. They were designed with intention. In her book, Mayor lists examples found in ancient Greek mythology on automatons. Though many were described as mindless, there were two exceptional groups described in *Iliad* and *Odyssey* that are ancient variants of AI. The first group were Hephaestus's helpers, "fashioned of gold in the image of maidens" and "bustl[ed] around their master like living women."[21] These golden assistants were not only mechanical servants, but were given human traits of consciousness, intelligence, learning, reason, and speech.[22] As a result, these Golden Maidens were capable of anticipating the needs of their human masters. Mayor argues that these golden assistants were artifacts of modern-day "augmented intelligence."[23]

[19] Mayor does, however, note that conceptions of artificial life have existed in ancient India and China as well.

[20] Adrienne Mayor, *Gods and Robots: Myths, Machines, and Ancient Dreams of Technology* 1 (2018).

[21] *Id.* at 149.

[22] *Id.* at 150.

[23] *Id.*

The second group were the Phaeacian ships that did not require "rudders or oars, no human pilots, navigators, or rowers, but are steered by thought alone."[24] Mayor notes that these ships were controlled by "some sort of centralized system with access to a vast data archive"[25] of the ancient world. Evidently, these vessels are clear parallels of current automated navigation systems. More importantly, Mayor reveals that, even in ancient Greek mythology, devices of artificial life took many forms. The aforementioned examples are perceived as assistive tools, extending the capabilities of the Greek gods and humans alike.

Interestingly, Mayor's text also highlights examples of technology as manifestations of tyrannical power. Talos, the bronze giant that was programmed to protect the kingdom of Minos, would spot strangers and hurl boulders to sink foreign vessels.[26] Talos was also built by Hephaestus, the Greek god of forge and patron of invention and technology, and commissioned by Zeus, the king of all Greek gods. In the very code of its being, Talos was made for destruction. Modelled after human traits, Mayor describes Talos perverting and reconfiguring the warmth of human embrace as a tactic for 'roasting' humans alive.[27] Talos was not the only device of merciless annihilation. Hephaestus also built Pandora. In contrast to the narrative most commonly known about 'her,' Pandora was, in fact, neither naïve nor a young woman. That is, Pandora was commissioned by Zeus to be made as a form of a revenge on humanity.[28] Her very design was purposefully measured with "gleeful malice toward the human race."[29] She was portrayed as a fabrication of evil disguised as beauty. Like Talos, she was programmed for the specific task of releasing sorrow and misfortune into the human world.

Beyond representing wickedness, Pandora was stunning. The gods were depicted as marveling at her human likeness.[30] Her beauty was captivating. The story of Pandora mirrors Pier Giuseppe Monateri's painting of the 'sublime' in *Dominus Mundi: Political Sublime and the World Order*. The aesthetic of the sublime is discussed as boundless, a dualism of fear and attraction. Though a

[24] *Id.* at 151

[25] *Id.*

[26] *Id.* at 7.

[27] *Id.*

[28] *Id.* at 156.

[29] *Id.* at 157.

[30] *Id.* at 158.

clear sign of imminent threat, the consciousnesses is submerged by the devilish trance and magnetism found in fear.

Across ancient Greek mythology, Mayor delineates, with great intention, between laborsaving devices and others that were "deliberately intended to inflict harm."[31] Nevertheless, both variations stand on the belief that machines are remarkable. These fictions are symptomatic of the pervasive charm of manufactured realism. Ultimately, Mayor nudges at lessons from ancient myths on the allure of the machine ushering in an idealization of imagined worlds.

In *the Age of Surveillance Capitalism*, Shoshana Zuboff describes the "mandate of prediction imperative,"[32] a pursuit of certainty that regards complete and total information as ideal. Machine intelligence becomes the restoration of "humankind to the Garden of Eden, lifting us from toil and struggle into a new realm of leisure and fulfillment."[33] The result: a utopia of certainty.

Zuboff explains that the desire for incontestable certainty and predictive utopia dates back to eighteenth-century imaginative thought on a rational systemic vision towards scientific techniques of forecasting.[34] These imaginations were then furthered in the early twentieth century by German experimental psychologist, Max Meyer. Meyer's prescription for modernity articulated a "scientific objectification of human experience and its reduction to observable measurable behavior."[35] Building on Meyer's vision, behavioral psychologist B.F. Skinner defined a utopia of technique and scientific dominion, substantiated in his novel *Walden Two*. In this text, Skinner outlines a community built on manipulating contingencies of rewards and punishments. Zuboff argues that these ideas have since been brought to life through the rhetoric of surveillance capitalism, an expression of Skinner's tools and imaginings of instrumentarian power and totality.[36]

She raises the sweeping impact of this utopia, falling under the radar of consciousness. She focuses on how technological practices appear to be theoretically agnostic and, instead, the 'magic' of and

[31] *Id.* at 128.

[32] Shoshana Zuboff, *The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power* Chapter Fourteen (2019).

[33] *Id.*

[34] *Id.* at 212.

[35] *Id.* at 349.

[36] *Id.* at 374.

fascination with machines capture humans in a state of awe.[37] Interestingly, Zuboff delves into the surveillance capitalist pursuit towards the collective mind and fantastical dreams of surrendering the individual for a shared knowledge.[38] Networks of machines operating in unison are a mirror to prospective human-machine relations, blurring the line between animate to inanimate and transforming relationships to objects interacting within the system.[39] The imposition of measured and automated rules are seamlessly integrated into societal operations.

The notion of formal indifference strikes a chord. Zuboff describes a "form of observation without witness," interpreting the intangible as measurable.[40] She notes that, in dehumanizing methods of evaluation, there is a reframing of equality to equivalence.[41] The seductive hum of the machine becomes the anthem of the techno-utopia.

A dichotomous process occurs where impenetrable complexity is met with simplification; a new signature and a "robotized veil of abstraction."[42] Undeniably, the integration of law in AI is an appeal towards the grid; a promise of "enduring and definitive charting of the legal world."[43] Legal concepts are further bound and placed in a distinct time and space. Clarity and consistency are reinforced by endless records and instructions such that the law may be "gapless, determinate, and nonoverlapping."[44] Furthermore, the migration away from social relations allows the legal actor to be "removed from responsibility for the worldly consequences of his actions."[45]

In the techno-utopia, "objectified computational behavioral metrics"[46] swallow human experience and thrive on ubiquity. Zuboff warns of the aspirational vision of surveillance capitalists for a complete system; one that is built and contained in a world of total knowledge. Knowledge becomes

---

[37] *Id.* at 382.

[38] *Id.* at 383.

[39] *Id.* at 384.

[40] *Id.* at 354.

[41] *Id.*

[42] Zuboff here is, of course, articulating a new mechanism of society. She describes a form of power derived from a way of knowing that dehumanizes qualitative means of evaluation and produces instead "equivalence without equality." She sees "objectification [as] the moral milieu in which our lives unfold." See *id.*

[43] Pierre Schlag, *Commentary: The Aesthetics of American Law*, 115 HARV. L. REV. 1047, 1055 (2002).

[44] *Id.* at 1059.

[45] *Id.* at 1060.

[46] Zuboff, *supra* 32 at 375.

both the currency and vessel of submission. As opposed to Mayor's imaginations from Greek mythology, Zuboff's text suggests that the surrender of humanity at the foot of the instrumentarian rule is imminent. In contrast to the willful draw towards the machine, Zuboff's painting of surveillance capitalism reflects a silent capture and descent into a vortex of quantifiable instruction.

Julie Cohen unpacks the notion of internet utopianism, reflecting on the burgeoning shifts and evolution of a society facing informational capitalism. While Zuboff provides a comprehensive illustration of this utopia, Cohen narrows the scope to the legal realm; how existing legal institutions must change to ensure rights and human freedoms are protected. She considers the double-edged sword of the open content model that has enabled the "emergence of new information businesses whose revenue models are based on harvesting and monetizing the data flows"[47] The internet and its "networked virtual spaces," she states, is perceived as "sites of utopian separation for the life of the mind."[48] Yet, the internet is evidently "embedded in real-world societies" that require real institutional solutions.[49]

What Cohen highlights then is the divorce between the virtual with the real. That is, the utopia is one that is imagined and not of the existing world. The problem is that there is no separation. The virtual space is built from the messiness of existing societal constructions. Consequently, the conceived distinction suggests that the existence of this utopia does not have implications nor effects on real-world institutions. Evidently, this fosters what Zuboff articulated as the lack of consciousness around the cooptation of a new methodological and quantitative tyrant.

Cohen, like Zuboff, suggests that the seed towards "control" and the instrumentarian reign has been long planted.[50] Automated information systems, that were introduced in the industrial-era, and constructed global networked supply chains, have circumvented institutional governance. In turn, transnational corporations with informational competencies have "nearly unlimited authority over their workers and outsize influence over the surrounding communities."[51] The introduction then of

[47] Julie Cohen, *Internet Utopianism and the Practical Inevitability of the Law*, 18 DUKE L. & TECH. REV. 85 (2019).

[48] *Id.* at 89.

[49] *Id.*

[50] *Id.* at 92.

[51] *Id.* at 93.

global platform businesses have merely capitalized and exploited the private economic power of an existent infrastructure.

As a result, data-driven, algorithmic processes only amplify obstacles around accountability.[52] The decisions produced by machine learning technologies cater to specificity, concealing reasoning and offering the impression as standalone end products. That is, they are considered themselves conclusive and representations of evidentiary analysis. Cohen argues that these technologies "sit in profound tension with traditional articulations [...] and commitment to the rule of law."[53] This erects barriers around judicial oversight, and in effect, unraveling fundamental rights. Evidently, Cohen's arguments point towards new modes of institutional governance that could confront networked informational systems that have long escaped traditional paths of accountability. So, what might these new modes look like? Frank Pasquale reflects on these questions in the *New Laws of Robotics*.

In his text, Pasquale explores the various ways in which AI has taken hold. In particular, he shifts away from the utopia/dystopia duality and, instead, reflects on the immediacy of attaining balance. Importantly, he stresses the role of AI as largely complementary and the ways in which this should be maintained as the path forward. In contrast to Cohen and Zuboff's bleaker, more cautionary tone, Pasquale offers a glimmer of hope around how humans can and must remain in reign of its machines.

As opposed to a (brave) new world, Pasquale introduces the four "new laws of robotics," an homage to science fiction writer Isaac Asimov's "Handbook of Robotics, 56th edition" in his short story "Runaround." These new laws are as follows:[54]

1. Robotic systems and AI should complement professionals, not replace them.
2. Robotic systems and AI should not counterfeit humanity.
3. Robotic systems and AI should not intensify zero-sum arms races.
4. Robotic systems and AI must always indicate the identity of their creator(s), controller(s), and owner(s).

For Pasquale, these four laws (principles) should be applied across all facets of society where AI may interfere. Fundamentally, the laws project a "humane agenda"[55] around the "strengthening of existing

[52] *Id.* at 95.

[53] *Id.*

[54] Frank Pasquale, *New Laws of Robotics: Defending Human Expertise in the Age of AI* 3-11 (2020).

[55] *Id.* at 4.

communities of expertise and the creation of new ones."[56] His argument centers around ensuring the resilience of human intervention; that technology cannot calculate out human beings. He distinguishes between "humanizing technology and the counterfeiting of distinctively human characteristics."[57] Evidenced in his language are his perceptions of a boundary between proper and improper integrations of technology. Technology that 'humanizes' will make processes more complex and further intellectual work. Replication, on the other hand, is an extension of simplification. It has the capacity to reduce and distill perceived messiness and uncertainty to a 'refined, perfected' form. Imitating 'humanity' and "falsifying features of actual human existence"[58] then dangerously depreciate human value.

Throughout his case studies, Pasquale reaffirms his four laws as the path forward to ensuring that technology will always be second to human guidance. Importantly, Pasquale further concretizes his argument but continually drawing examples from existing technological use. As opposed to descending into prospective dystopic visions, he is focused on the present and near future. This is particularly powerful statement as he reconciles "science fiction," media and cultural portrayals of AI, with actual use. Moving from imagination, Pasquale brings AI to the ground.

Perhaps the most important of his four laws is the last: ensuring a path of responsibility between human to machine. There again, Pasquale delineates between depictions of AI and their actual practice. As opposed to having lost control of the robots,[59] he traces the line of responsibility and how accountability is transferrable from one person or entity to another.[60] The significance of this fourth law is that the human is never lost, and especially in the face of liability. More importantly, he reaffirms the need for a realignment of values. How the human is to remain in-the-loop is a reconceptualization of professionalism and expertise. The former, he argues, involves the "recurrent need to deal with conflicts of values and duties."[61] The latter builds on this notion. That is, professionalism should account for expertise that "cannot simply be reduced to equations of

[56] *Id.*

[57] *Id.* at 7.

[58] *Id.* at 9.

[59] Pasquale alludes to the fantastical imagination of the robots that develop their own conscience (i.e., HAL), and distinguishes from unforeseen consequences or unintended results. See *id.* at 12.

[60] He cites how programmers may be held responsible for building in certain constraints, but an entity that then disables these constraints should be held responsible. See *id.*

[61] *Id.* at 19.

efficiency and algorithms of optimization."[62] In short, Pasquale argues for the safeguarding of human values, democratic representation, and social goals. Consequently, the translation of tasks into code is not purely technical. For Pasquale, it is an "invitation to articulate what really matters in the process."[63] So, what really matters in law?

Systems Alignment and Philosophical Aspirations

Turning to the legal system, Benjamin Alarie contends that technology pushes forward the law by bridging gaps of indeterminate legal standards with precise rules identified by AI.[64] He articulates that a combined increase in "observable phenomena" and heightened accuracy in pattern recognition technology will lead to the "legal singularity."[65] For Alarie, this is the path of the law. The notion of 'legal singularity' draws from an association of the law as precise, predictable, and certain in its function.[66] The underlying view is that principles of the law, in its present form, lack certainty. AI aids with the crystallization of the law, clarifying existing principles by reinforcing standards as rules. AI then would bring certainty out of specificity. In effect, legal indeterminacy is perceived as a threat; a tell that the law's current state is one of incompleteness.

Alarie regards the incompleteness of the law as a weakness of the system. He argues that the over- and under-inclusiveness, as a result of being incomplete, has subsequently led to exploitation of the system. Fortunately, he notes that the legal singularity will bring about the "elimination of legal uncertainty and emergence of a seamless legal order, universally accessible in real-time."[67] The law will achieve functional completeness.[68] The vision of legal singularity is, of course, reminiscent of the techno-utopia. It is the perception that a gapless grid and quantitative alignment resolves the existing

---

[62] *Id.* at 23-24.

[63] *Id.* at 28.

[64] Benjamin Alarie, *The Path of the Law: Towards Legal Singularity*, 66 U. TORONTO L.J. 443, 445 (2016); *see* also Benjamin Alarie et al., *Law in the Future*, 66 U. TORONTO L.J. 423, 427-28 (2016).

[65] *Id.*

[66] *See* Theories of Adjudication, in particular the discussion on stare decisis as the 'life blood of legal systems,' requiring precision in addition to stability and certainty. Michael Freeman, *Lloyd's Introduction to Jurisprudence*, (9th ed. 2014).

[67] Alarie, *supra* 64 at 445.

[68] *Id.*

unpredictability in the legal system. He argues that machine learning technologies allow the removal of emotion, providing unified classifications through objective and logical operations.[69]

Alarie notes that "data and better machine learning inference tools are likely to be complements to human judgment rather than substitutes."[70] He suggests that experts will work with big data and machine learning technologies to elevate certainty in the performance of legal work. He describes how reliance on big data and machine learning models to inform decisions will "optimize" the content of the law. The implication is that machines are capable of identifying "what the law should be in order to achieve our implicit social objectives."[71] Again, for Alarie, the law is incomplete owed to "limited data and imperfect information."[72] As a result, provided that the legal system has yet to achieve equilibrium, further developments in machine learning tools will eventually shift the role of machines as complementary to machines as substitutive. Ultimately, arriving at the legal singularity will be inevitable.

Alarie's vision of a legal techno-utopia provides a rather one-dimensional perspective in the sphere of technological integration. In classic law and economics fashion, his arguments stem heavily from notions of optimization, equilibrium, and efficiency. Moreover, Alarie conflates legal with machine complexity. In turn, complexity is loosely referred to as the competence to process information and provide a decision. Consequently, 'computing power' appears as a rather suitable substitute with statistical inference absorbing human reasoning. Law is now perceivably computation.

Perhaps in direct response to Alarie,[73] Christopher Markou and Simon Deakin ask the question of whether the law is indeed computable. Their initial reaction speaks to the inherent normativity of the legal system. In particular, Markou and Deakin raise the perspective that 'obedience,' or compliance, is not guaranteed.[74] That is, the legal system necessarily depends on an anarchic component that enables an introspective evaluation. In effect, 'scrutiny' allows for checks and

---

[69] *Id.* at 450.

[70] *Id.*

[71] *Id.* at 453.

[72] *Id.*

[73] Markou and Deakin specifically cite "the boldest vision" and legal singularity. *See* Christopher Markou and Simon F. Deakin, "Is Law Computable? From Rule of Law to Legal Singularity," University of Cambridge Faculty of Law Research Paper (Apr. 30, 2020) 5, available at: https://ssrn.com/abstract=3589184.

[74] They cite H.L.A. on "question of obedience" and "its demands must in the end be submitted to a moral scrutiny." *See* H.L.A Hart, *The Concept of Law* 210 (3rd ed. 2012). See *id* at 7.

balances that maintain the dynamics of power and legitimacy. Nevertheless, Markou and Deakin trace the origins of computational fervor as attributable to Gottfried Wilhelm Leibniz and the realization of his mathematical dream.

Markou and Deakin describe how Leibniz was enveloped in creating a universal language, capable of reducing all reason to logical calculus.[75] Accordingly, they suggest that Leibniz's framework to "formaliz[e] human thought with logico-mathematical calculations" became the "precursor to the development of computer science."[76] Putting his theory to test, Leibniz chose law. He perceived law as a rational framework for organizing society. As a result, Leibniz was convinced that his model would further heighten the precision of legal rules through axiomatic reduction.[77]

Advancing through the historical developments of the common law,[78] Markou and Deakin reflect on the subtle remnants of Leibniz's axiomatic method. They argue that the current generation of AI-assisted Legal Tech rests on Leibniz's assumptions of a "purified essence to law and legal reasoning" capable of "mathematization."[79] Therefore, the deductive approach "accomplishes little more than ossifying legal concepts into self-evident computational 'truths.'"[80] Perhaps most powerfully stated is their argument that Leibniz's method results in a simplification of the "legal ontology that assumes these concepts are stable referents."[81]

Markou and Deakin then confront Alarie's vision of legal singularity from the perspective of complexity. That is, machine complexity is not legal complexity; and an increase of the former subsequently leads to a decrease in the latter. In short, they argue that the law is not computable, as the "binary nature of computation means that all legal problems must ultimately be decidable using binary logic."[82] Though Markou and Deakin provide convincing arguments around the incommensurability of law and Leibnizian binaries, they perhaps ironically treat law and computation as a binary. The duality they argue against is precisely their perceived approach in

[75] *Id.* at 11.

[76] *Id.* at 12.

[77] *Id.* at 12.

[78] They consider the competing schools of thought between formalism and legal axioms and realism. See *id.* at 14.

[79] *Id.* at 18.

[80] *Id.*

[81] *Id.*

[82] *Id.*

interpreting the implicit goals of Legal Tech. Importantly, computer science methodology and legal reasoning extend far deeper, and in a more nuanced manner, than they describe. That while the law embodies an open texture and is incomplete, it equally relies on logic and should not be dismissed. This means that as opposed to a systemic level analysis, understanding the computability of law requires a more granular approach.

Therefore, I contend that analysis should be conducted at a micro-level, and specifically to the granularity of linguistic deconstruction. Furthermore, I argue that the particularities of the law have been captured in its specific technical language. As a result, a shift from natural language to code – or a migration of mediums – necessarily reveals the impact of computation in law. Moreover, it offers opportunities to reflect on whether they are, in fact, incommensurable, or that there may be space for reconciliation. Nonetheless, it may be important to clarify specifically what the definitions and parameters of AI and law are. To do so, we shall turn to the law's encounter with AI.

When Law Met AI

When discussing AI and law, to what does it refer? Harry Surden provides an incredibly helpful and thorough account of the various forms in which AI has taken shape, particularly in the legal space. Echoing Pasquale, Surden draws attention away from speculative discussion and towards the law and policy issues raised by AI technology today.[83] To start, Surden defines AI as the use of technology to automate tasks that involve human intelligence.[84] Surden further refines the definition to specify human intelligence as requiring "cognitive activity."[85] He is careful, however, to distinguish cognitive activity from synthesizing human-level thinking. Surden intentionally focuses on current[86] AI technology. This includes systems that rely on heuristics; otherwise, the use of certain computational approximations that help identify "discernible underlying patterns and structures."[87] In effect, these include machines that appear to do the work that typically requires human cognition. This differs

[83] Harry Surden, *Artificial Intelligence and Law: An Overview*, 35 GA. ST. U. L. REV. 1305, 1306 (2019).

[84] *Id.*

[85] *Id.*

[86] Surden specifies the "near-term time frame" of 5-10 years roughly. See *id.* at 1308.

[87] *Id.* at 1309.

from what is known as Artificial General Intelligence (AGI), or "thinking machines with abilities to meet or surpass human-level cognition."[88]

Surden raises two AI approaches that most commonly are featured in the Legal Tech space: (1) machine learning; and (2) logical rules and knowledge representation.[89] Importantly, Surden provides a clear outline of the type of work these two approaches are capable of and can enable. With machine learning, Surden is careful in clarifying the meaning of learning. He stresses that 'learning' is a "rough metaphor"[90] and is effectively a quantitative proxy, or a 'functional' understanding of learning. Machines then 'learn' in the guise of 'progress,' by examining data and searching for patterns.[91] Subsequently, their performance improves through the introduction of more data and the refining of these patterns.

To substantiate his definition, Surden applies the helpful example of machine learning systems identifying "spam" emails. These systems are capable of automatically detecting emails that are unsolicited through various "signals."[92] These signals provide a strong likelihood that the email is spam. In this case, the signals could include word probabilities (i.e., presence of a particular word, email origin, etc.).[93] With increasingly powerful models of machine learning, Surden expresses that this approach in AI offers incredible insight. Nevertheless, its data-dependence offers limitations around its current competencies in the legal space.

Alternatively, expert systems, or logic rules and knowledge representation, "model real-world phenomena or processes in a form that computers can use, typically for the purposes of automation."[94] As revealed in its name, expert systems involve providing computers a set of rules that "represent the underlying logic and knowledge"[95] of the activity being modelled. These rules must

[88] *Id.* at 1308.

[89] *Id.* at 1310.

[90] *Id.* at 1311.

[91] *Id.*

[92] *Id.* at 1314.

[93] *Id.* at 1313-1314.

[94] *Id.* at 1316.

[95] *Id.*

be written in a computer-understandable form, as they behave as instructions for computers to process information. How this information is processed typically follows a deductive logic.

In order for knowledge-based AI systems to 'reason,' software developers must work in consultation with experts; in effect, translating the meaning and logic of a specific area of expertise to a "set of comparable formal rules."[96] Rules-based knowledge representation systems must define, in advance, both operating and decision rules.[97] However, this is not to suggest they are less complex than machine learning systems. Instead, computers are capable of manipulating these predefined rules in "deductive chains to come to nonobvious conclusions about the world."[98] Knowledge-based AI systems, then, can combine facts and apply logical rules to arrive at conclusions that may be difficult for humans to discern.[99] Moreover, though they are frequently regarded as two separate approaches, complex systems could involve hybrids of these systems. This enables the strengths of each approach to tackle specific tasks.

Surden cautions that AI is effective for tasks that either (1) involve "clear, unambiguous rules,"[100] or (2) have rather identifiable "underlying patterns or structure."[101] Where there may be abstract concepts that cannot be meaningfully encoded, AI technologies do not perform well. Equally, tasks that involve subjective interpretation, or social choices, tend not to be suitable for AI automation. So, what might be the role for AI in law? He describes AI and law as the "application of computer and mathematical techniques to make law more understandable, manageable, useful, accessible, or predictable."[102] According to Surden, the use of AI in the legal field impacts three categories of users: (1) practitioners; (2) administrators; and (3) those governed by the law.

For tasks traditionally performed by lawyers, document review and litigation discovery are common candidates of automation. He argues that these types of tasks are routine, "*mechanical* and repetitive" in nature.[103] For tasks traditionally involving administrators of the law (i.e., judges and government

[96] *Id.* at 1317.

[97] *Id.*

[98] *Id.* at 1318.

[99] *Id.*

[100] *Id.* at 1323.

[101] *Id.* at 1324.

[102] *Id.* at 1327.

[103] *Id.* at 1331.

agencies), Surden considers the use of algorithms for "risk-assessment scores," particularly on likelihood of recidivism, and the assessment of government benefits programs.[104] The former usually draws on machine learning technologies and past crime data, while the latter is knowledge-based and involves modelling the rules used to 'calculate' benefits. In both scenarios, their outcomes can influence the decisions of the administrators and can be problematic as there may be biases that are "encoded" in these technologies. As well, it is important to note that these are not the only types of technology used, or considered, in these settings.[105] Finally, the third category involves "users of law."[106] Surden categorizes these technologies as tools that are helpful in providing insight into various aspects of the legal system. These include computable contracts and "legal self-help systems."[107] The former is defined as legal contracts that may be expressed in a computer-understandable form. The latter are "simple expert systems" that provide "answers to basic legal questions."[108]

In short, Surden provides a strong overview of the various approaches to AI, and particularly in the legal field. Moreover, he offers a concrete discussion, shifting away from idealistic imaginations. Therefore, it may be worth diving deeper into the types of skills that AI will impact in the legal industry, provided the continued integration of these technologies.

Mark Fenwick and Erik Vermeulen describe lawyers of the future operating as "transaction engineers." They argue that an increase in the uptake of AI-driven legal tools would render traditional skills of contract drafting, revision, legal risk management, and even dispute resolution obsolete.[109] This may be envisioned as the subcontracting of legal 'grunt work' to machines while humans are dealt the important tasks – in a sense, a Siri for law. Rather than a loss of skill, it is a

[104] *Id.* at 1333.

[105] The advent of 'online courts' has introduced the notion of virtual or Zoom courtrooms and asynchronous judging. These technologies are not quite within the realm of AI, but more in consideration of the significance of courts being physical. *See* Richard Susskind, *Online Courts and the Future of Justice* (2019).

[106] Surden, *supra* 83 at 1334.

[107] *Id.* at 1335.

[108] *Id.*

[109] Fenwick and Vermeulen expand on their idea of lawyers as the "transaction engineer;" effectively facilitators or 'project managers' in the deployment of new AI-driven legal technologies. Their argument suggests that lawyers will increasingly migrate into technology-based roles, working as middlemen between professions, that require comprehension of data analytics and computer coding. *See* Mark Fenwick and Erik Vermeulen, "The Lawyer of the Future as 'Transaction Engineer:' Digital Technologies and the Disruption of the Legal Profession," in Marcelo Corrales, Mark Fenwick, and Helena Haapio (eds.) in *Legal Tech, Smart Contracts and Blockchain* 256, 268-270 (2019).

reclassification between analytical and menial work. It is the subordination of certain skills in the name of efficiency and accuracy.

These ideas have previously been expressed in the literature on the disruption of legal practice and future of the legal profession.[110] Interestingly, in *AI for Lawyers*, Noah Waisberg and Alexander Hudek explore how AI has 'amplified' legal skills and expertise. Waisberg and Hudek provide a comprehensive overview of the ways in which AI can and should be embraced in legal practice. Moreover, they consult experts of the legal industry to provide an insider perspective on the concrete impact AI has had thus far. Not only are multiple chapters written by those who are founders of legal AI startups, they feature other industry leaders that have chosen to integrate these technologies into their internal legal departments.

In having a rock star cast, the text behaves as an empowering self-help book, providing a guided and practical approach on how the legal profession is transforming. The book is heavily case- based, with testimonials that offer the impression that the legal field indispensably depends on these technological insights. As Waisberg and Hudek are, themselves, leaders in the Legal Tech environment – having built one of the most powerful systems of contract review –it is difficult not to be drawn into the fervor.

Perhaps one of their most striking chapters addresses specifically the shifts in legal skills that Fenwick and Vermeulen discuss. Applying the Jevons paradox,[111] Waisberg and Hudek describe an increase in the efficiency of delivering legal services that will, in turn, expand and grow the legal field. Unlike Fenwick and Vermeulen, Waisberg and Hudek consider how legal knowledge will take hold and become "scalable."[112] In particular, their argument reflects on the bottling of legal knowledge and transferring it to technology. This includes the management of legal data (e.g., court-generated data, patent and other intellectual property data, data from case management systems) and those

[110] *See* Richard Susskind, *The End of Lawyers Rethinking the Nature of Legal Services* (2010); Albert H. Yoon, *The Post-Modern Lawyer: Technology and the Democratization of Legal Representation*, 66 U Toronto LJ 456 (2016); Brian Sheppard, *Incomplete Innovation and the Premature Disruption of Legal Services*, 2015 MICH. STATE L. REV. 1797 (2016).

[111] The notion that "when a resource is delivered more efficiently, the consumption of that resource will actually increase." *See* Noah Waisberg and Dr. Alexander Hudek, *AI for Lawyers* 22-26, 51-52 (2021).

[112] Waisberg and Hudek consider on how more work can be done with fewer resources (specifically by training machine learning models to perform legal work). See *id.* at 30, 71-80.

constructing the models to train systems to reflect the legal work and processes.[113] This suggests that while the legal profession may change, it fundamentally will remain a knowledge-driven industry. The question becomes how the use of legal information and representations of legal knowledge develop their own standards[114] of accountability and transparency.

As raised in the aforementioned section, there are a number of philosophical implications in the integration of computation with law. Even across the legal community, there is disparity in the underlying regard for the legal system. These disparate visions translate and embed[115] themselves into the technology. As a result, legal knowledge may become no less opaque for those seeking access, as information asymmetries are merely transferred from human to machine. Though the focus of the text implies how AI impacts specifically the parameters and skills required of legal professionals, the pending transformation[116] suggests an expansion of the field to those who may not have legal training. Consequently, the priority should not rest on efficiency of delivery, but instead, on determining methods of enabling deeper understandings of legal mechanics for its representation.

In a recent article, Joshua Browder suggests how code can increase the transparency, scalability, and equity of the legal system.[117] He reflects on the "lawyerly protectionism"[118] that has over time shielded individuals from accessing legal expertise. He argues that a "software-first approach"[119] can improve the current barriers that hinder most low-income individuals from legal assistance. Browder uses, as an example, the application process involved with claiming asylum status. He states that software has the capacity to embed legal knowledge in the intake form, such that legal information typically "hidden" from the public may be explicitly understood.[120] Another example he alludes to is the hosting of laws on an open platform. He considers how Washington DC's City Council has their laws available publicly on the software platform, Github. This allows residents to spot errors and

[113] *Id.* at 53.

[114] I note that while there are recommendations around how to train AI and who to consider when training, it is of Waisberg and Hudek's own advice. Moreover, defining subject matter expertise is an equally complex task that is left unanswered. For the "three keys to successfully amplifying learning through AI," see *id.* at 74.

[115] Waisberg and Hudek state, "capture legal processes in expert systems...embedding legal knowledge into systems and processes in order to automate aspects of legal work." See *id.*

[116] They describe it as "verge of a transformation in how lawyers see themselves and their roles." See *id* at 65.

[117] Browder, *supra* 2.

[118] *Id.*

[119] *Id.*

[120] *Id.*

submit instantaneous requests for change.[121] In publishing laws publicly and freely, citizens are able to review legislation and discover any "loopholes and special interests."[122] Ultimately, he points to the potential of software democratizing the law.

Browder provides a convincing case in how code is capable of bridging legal knowledge to the public. The examples he provides are perceivably "building blocks" towards a broader vision of the legal system as the operating system of society.[123] Evidently, these arguments reinvigorate conversations around the law/code dialectic. The question becomes whether these individual instances substantially demonstrate that law and code are interchangeable systems. It is then imperative to revisit Lawrence Lessig and the notion of code as law.

Legal Design and Law/Code Dialectic

For Lawrence Lessig, the conceptualization of code as law is not novel but rather intuitive. He draws attention to code as a form of control in the 'cyberspace;' that "code writers are increasingly lawmakers."[124] The difficulty, of course, is defining the parameters of the cyberspace. Lessig relays an interesting example of a dispute that unfolds in the virtual and in the real. In the real, it is perceivably a horrific event, whereby two neighbors, Martha and Dank, engage in a conflict over the death of Dank's dog. The dog had mistakenly consumed the poisonous flowers from Martha's garden. One of the particularly striking (and even peculiar) responses from Martha was her attempt to attribute fault to Dank for having a dog that suffered when it died.[125] This came as a reaction to Dank, questioning why poisonous flowers were being grown in Martha's yard in the first place.

In near seamless fashion, Lessig changes gears and paints this same dispute in the virtual. Rules and norms in the virtual seem to shift in a manner that mitigate the 'horror' of this neighborly conflict. Lessig suggests that through simple adjustments of the code, Dank's dog could die without suffering; or the poisonous flowers would become harmless if they were accidently blown off Martha's property.[126] The "'what happens when' is a statement of logic; it asserts a relationship that is

[121] *Id.*

[122] *Id.*

[123] *Id.*

[124] Lawrence Lessig, *Code 2.0* 79 (2nd ed. 2006).

[125] *Id.* at 13.

[126] *Id.* at 14.

manifested in code."[127] It appears, then, that the events of the virtual do not carry the same consequences as they do in the real.

Lessig, therefore, raises the problem of how the virtual translates the real. He asks, "what does it mean to live in a world where problems can be coded away?"[128] It follows, what is the relationship between law and code when the boundary between virtual and real is ill-defined? Though Lessig's example is rather simplistic, it poses an intriguing thought experiment around the meaning and implication of constructed laws. In the real, Lessig likens certain elements as definable, with choices that can be made and controlled.[129] These norms are understood as "man-made." In the virtual, everything is capable of being controlled through design and the construction of code. Can an analogy be drawn between law and code? What might be the differences, and are they significant?

To answer these questions, Lessig raises another provocative example. He compares computer "worms" with search warrants.[130] Provided that the computer worms can stay dormant until they are "activated" for a specific task, Lessig compares a computer worm with a warrant to search a citizen's premises. Search warrants are generally not authorized unless there is sufficient reason to breach a citizen's private property. Lessig considers whether a worm that may be designed to search through a citizen's computer can be likened to a search warrant. Moreover, he reflects on whether it is constitutional in accordance with the Fourth Amendment.[131] What Lessig highlights is, again, the complexity involved when legal instruments understood in the "real" space are performed in the "virtual." In this case, the notion of "search" using computer code introduces ambiguity around its permissibility. He classifies this form of ambiguity as latent ambiguity; in effect, expressing how code performs in a manner that reinvigorates questions of the intent and purposes of law.

Returning then to the story of thorny neighbors, Martha and Dank, Lessig argues that the shift from law to code is, effectively, structural. Regulation is enabled "by the very architecture of the particular space;" and that "its architecture will affect whether behavior can be controlled."[132] Consequently,

---

[127] *Id.*

[128] *Id.* at 15.

[129] *Id.* at 11.

[130] *Id.* at 20.

[131] Lessig considers whether the Fourth Amendment is protecting against burdensome 'invasions' of privacy, or generally, "suspicionless governmental invasions." See *id.* at 21-22.

[132] *Id.* at 24.

what Lessig introduces is the notion that certain structures are more conducive to the types of control enabled by the instrument; and, that it is irrespective of the space. This means that the virtual merely reopens the definition of existing forms of regulation but should not be treated as different from the real. In turn, the architectural construction could encourage some forms of control over others. Unlike Cohen, Lessig does not find that there are complexities of translation when shifting between the virtual and real. Alternatively, Lessig identifies the problem as whether instruments of control, traditionally performed via 'analog' law, can be performed using computer code. The question becomes: is control akin to regulation? If so, should code be law?

Interestingly, Alex Pentland reflects on how the law is, itself, an algorithm. He considers how "most laws and regulations are just algorithms that human organizations execute."[133] As a result, laws are inherently capable of translation given their code-like structures. He describes this as explanatory of the rising use of computers to assist and automate legal work. Nevertheless, Pentland argues that in order to harness the potential of 'legal algorithms,' there must be oversight and accountability mechanisms in place.[134] He suggests this requires, to an extent, modularization. That is, the design must account for both humans and software working in tandem towards the goals of the system.[135] Modularity ensures that systems may be tested and evaluated continuously to ensure they are adaptive to the circumstances of its environment. In the case of legal systems, it must continually reflect legal processes. What Pentland articulates is then a hybrid architecture whereby computational tools may be integrated with human intervention.

Pentland outlines five components he finds currently missing in order for 'computational law' to be successful. These include: (1) specification of system performance goals; (2) measurement and evaluation criteria; (3) testing; (4) robust and adaptive system design; and (5) continuous auditing.[136] These five elements suggest that the legal system is not currently equipped to provide for "good governance."[137] These components may be regarded as useful markers. Though, it may be argued that only the first – the specification of system performance goals – is of concern. The question is

[133] Alex "Sandy" Pentland, *A Perspective on Algorithms*, MIT Computational Law Report Release 1.0 (2019), available at: https://law.mit.edu/pub/aperspectiveonlegalalgorithms/release/3.

[134] *Id.*

[135] *Id.*

[136] *Id.*

[137] *Id.*

whether the objectives of legal systems can be articulated such that measurement criteria would follow. To answer this question, we must necessarily turn to Mireille Hildebrandt.

In *Smart Technologies and the End(s) of Law*, Mireille Hildebrandt reflects on the architectural structure of legal systems and whether they may be reconcilable with technological design. She argues that applying code as law, or regulation with technology, would lead to the end of law.[138] Hildebrandt distinguishes between the idea of 'legal by design' (LbD) with Legal Protection by Design (LPbD). The former is a "subset" of techno-regulation; these technologies have a "*de facto regulatory effect*."[139] These regulatory effects may be deliberate or the result of unforeseen consequences. Importantly, LbD requires two specifications: (1) an unambiguous interpretation of the relevant legal norm; and (2) translation of the interpretation to a programming language.[140] She argues that the goal of LbD is compliance, owed to the rigidity of computer code.[141]

Pioneering alternatively the notion of LPbD, which she further elaborates in her seminal text, *Law for Computer Scientists and Other Folk*, legal norms must be accommodated in the design requirements to properly align with socio-technical innovation. LPbD is understood as maintaining the integrity of "legal" in the context of fundamental rights. This means that "the scope of LPbD should be determined by way of *democratic participation*," and the ability to "*contest its application in a court of law*."[142] Hildebrandt suggests that, unlike other forms of "ethical requirements" that are integrated in the technological design, the choice architecture, under the requirements of LPbD, is not subjected to market forces nor the creators' own ethical predispositions. This ensures that, structurally, the protections afforded by 'analog' (enacted) law are upheld.

Moreover, LPbD applies a method of 'resistability,' the capacity to 'rule out' deterministic environments.[143] Ultimately, LPbD is the assurance that technological norms do not overtake legal norms. Consequently, the missing component of "system performance goals" articulated by Pentland is not, in fact, missing. Rather, these performance goals are precisely the goals of "justice,

[138] Mireille Hildebrandt, "The end of law or Legal Protection by Design," in *Smart Technologies and the End(s) of Law: Novel Entanglements of Law and Technology* 214 (2015).

[139] Mireille Hildebrandt, "'Legal by Design' or 'Legal Protection by Design'" in *Law for Computer Scientists* 267(2020).

[140] *Id.* at 268.

[141] *Id.* at 268.

[142] *Id.* at 269.

[143] Hildebrandt, *supra* 138 at 218.

legal certainty, and purposiveness,"[144] norms that have always underpinned the legal system. Hildebrandt reinforces that the other components, including testing and measurement criteria, should all pivot around compatibility with legal norms. The point of departure, she states, is the task of bridging legal with computational across all facets.[145] It follows that the systems' mechanisms should become the focus of the study.

I have previously discussed that, as opposed to systems-level alignment, a turn to a more granular investigation is necessary. In recent years, the fields of legal analytics and legal informatics[146] became of particular interest. Kevin Ashley reflected on how the 'open texture approach' of early argument retrieval and cognitive computing systems laid the foundations for computational models of legal reasoning.[147] Though he argues that law is composed of rules, Ashley states that features of vagueness and the open texture of statutory provisions need to be addressed. Therefore, he reflects on the significance of legal text. In particular, he analyzes how the methods of legal reasoning are centered around complexities associated with semantic and syntactic ambiguity. As a result, issues of translation emerge when using computational tools to model legal reasoning.

His proposition, alternatively, is to further the cognitive computing paradigm by heightening practices of legal information retrieval. Like Surden, Ashley clarifies that cognitive computing does not involve building intelligent systems to 'think' nor to provide a solution to the user's problem.[148] The intention is for the human to tailor the information relevant for a specific task. This means that the human must indicate in advance the specific knowledge and concepts they would like the machine to identify. Unlike expert systems, cognitive computing does not depend on the specification of rules. Rather, it is the gathering of rules from relevant knowledge. In this case, cognitive computing systems regard legal knowledge as "embodied in the corpus of texts from which the program extracts candidate solutions or solution elements and ranks them in terms of their relevance to the problem."[149] What may be gathered is that legal knowledge is preserved and found

[144] *Id.*

[145] *Id.*

[146] Broadly understood as the use of information technologies and data to drive insights in the legal field. For further detail, *see* Dan Martin Katz, Ron Dolin, and Michael J. Bommarito (eds), *Legal Informatics* (2021).

[147] Kevin Ashley, *Artificial Intelligence and Legal Analytics: New Tools for Law Practice in the Digital Age* 11-13 (2017).

[148] *Id.* at 12.

[149] *Id.* at 13.

within the words of legal texts. In effect, in developing methods of analyzing legal language, we may be able to reconcile law with computation. Furthermore, it suggests that understanding the linguistic patterns of legal language provide stronger tests around the limits of legal computability.

Accordingly, we return to Hildebrandt who provides an astute account around law as driven by text. In her recent article, "The adaptive nature of text-driven law," she identifies how normativity is enabled by the semantic ambiguity inherent in natural language. This suggests that the adaptive nature of legal norms is afforded by the flexibility of meaning. Legal norms then necessarily require the "open texture of natural language."[150] In contrast, "code-driven law" resists contestability and exchanges legality[151] with legalism. This is owed to mistaken assumptions around disambiguation as a proxy for legal certainty. Instead, she argues that existing mechanisms of the legal system already account for multi-interpretability. This means that legal certainty is not an issue that demands resolving. Therefore, the "over- and under-inclusiveness" associated with "disambiguated computer code"[152] actively removes legality from the law. Consequently, priority should remain with text-driven law and computational technologies that "challenge unwarranted legalism."[153]

Hildebrandt, then, puts forward a test: is natural language the only vessel in which legal norms may be housed? It is, thus, on this premise that I conduct the remainder of my dissertation. In the following chapter, I reflect on the long-standing and intimate relationship between law and language. It is there where I shall reopen the inquiry around the characteristics and uniqueness of the legal language.

[150] Mireille Hildebrandt, *The adaptive nature of text-driven law*, J. OF CROSS-DISCIPLINARY RESEARCH IN COMPUTATIONAL LAW (CRCL) 1,10 (2020).

[151] Legality she describes as the combination of justice, purposiveness, and legal certainty together. Legalism is the prioritization of legal certainty over justice and purposiveness. See *id.*

[152] *Id.*

[153] *Id.*

1- The Linguistic Affair

Understanding the "Law of Interpretation," or better, how to reason with legal texts is one of the most fundamental and oldest questions in legal practice. Some legal scholars consider a theory of legal interpretation as one founded on the premise that legal norms exist within the words of the page.[154] That is, the limits of the text are the limits of the law.[155] This suggests that legal interpretation is necessarily a linguistic matter.

Since the 1960s, the structures of written legal language had been analyzed in depth.[156] However, an exploration of the symbiotic relationship between law and language did not appear until the 1970s. For it was Brenda Danet, in "Language in the Legal Process," who reflected precisely on legal language and its role in the ordering of social relationships. She argued that language is the medium through which the law does its work.[157] Language is the law's functionary.

The relationship between law and language has always been one born of necessity. Language is often conceived as the vehicle in which legal norms could embed itself, the house but not the home. Consequently, language is important to the law, but only as a tool through which the law is realized. The underlying assumption is that law and its language exist in a state of universality and is logically reducible. Most fascinating, though, is the belief that description is distinct from interpretation; that in describing the law, the language is seen as quantitative and objectifiable. Yet, the law hinges on social and political metaphors that require latent understanding of temporally specific societal constructs. These complex relations and interactions are then encased and deployed in a technical grammar. This begs the question: is the medium the message?

As may be inferred, this chapter revisits the seminal conversations around law and language, walking through the perspectives of leading scholars that have highlighted the unique behaviors of legal language.[158] Through the voices of these scholars, I will attempt to weave the undercurrents of law and language as presented in the realms of legal, linguistic, and literary theory, as well as the

---

[154] *See* for example, Vittorio Villa, *A Pragmatically Oriented Theory of Legal Interpretation*, 12 REVUS: J. FOR CONSTITUTIONAL THEORY AND PHILOSOPHY OF LAW 89 (2010) available at: https://journals.openedition.org/revus/146.

[155] A. Barak, *Purposive Interpretation in Law* 6-7 (2005).

[156] David Mellinkoff's text investigates the specific usage of written legal language, analyzing the structures of various origins including Latin, French, and Anglo-Saxon. *See* David Mellinkoff, *The Language of the Law* (1963).

[157] Brenda Danet, *Language in Legal Process*, 14 L. & SOC. REV. 445 (1980).

[158] This chapter recognizes there are limits to the comprehensiveness of the arguments, that there may be far more to add to the conversation. Nevertheless, the chapter is a best-effort and serves to be introductory.

philosophy of language. This chapter serves as a break from the digital encounter to return to the roots of language as a frame of analysis. Methodologically, the section follows existing tensions surrounding legal interpretation. Namely, three key debates will be considered: (1) the difference between clarity and precision; (2) the paradox of form and substance; and (3) the myths of the fact-law distinction. In observing these debates, the aim is to provide insight into the mysteries of legal writing. More importantly, they help uncover the role of legal language in law's interpretative exercise.

The Language of Law

There is then no better place to start than the work of David Mellinkoff. In the preface of Mellinkoff's pioneering text, *The Language of the Law*, he highlights a quote by legal historians on the significance of language. That is, language is "no mere instrument which we can control at will; it controls us."[159] This sets the tone of his work, noting that law has been a subject of its tool. His text is a first of its kind, a systematic examination of language in legal text. Language, he states, is not only intended to express, but also to convey thought. This distinction between expression and conveyance is particularly fascinating. He suggests that communication necessarily requires both components. Is expression merely stylistic and is the function of legal language purely communicative?

In his text, Mellinkoff advances a veiled historical account on the development of legal language; ultimately, culminating to his conclusion that the language of the law should not differ from common speech. He introduces his argument by defining the boundaries and characteristics of language in law. His text is subsequently divided into two parts: (1) how the language has come to be; and (2) how it is being used. Though interesting, I shall focus on his arguments on the latter for the intentions of this chapter. Mellinkoff states that this "customary language" used by lawyers and legal scholars includes a distinctive vocabulary, "certain mannerisms of compositions,"[160] and legalistic jargon, and words imported from other languages such as Latin and French. He argues that the combination of these factors has led to the divergence between the language of the law and ordinary language. More specifically, he outlines nine characteristics that have made legal language a "specialized tongue."[161]

---

[159] Mellinkoff cites a number of works, *see* specifically 1 Pollock and Maitland, *The History of English Law* 87 (2d ed. 1898), Oliver Wendell Holmes, *The Common Law* 382 (1881), and 2 Bacon, *Works* 192-193 (Montagu ed. 1825).

[160] Mellinkoff, *supra* 156 at 3.

[161] *Id.* at 11.

Amongst these characteristics, the majority draw attention to the vocabulary. For Mellinkoff, the words of the language are problematic; a recurrent theme that these terms of art are a significant source of confusion. Consider the first characteristic he discusses: the frequent use of common words with uncommon meanings. He highlights that words understood by the lawyer are "incomprehensible" to those outside the community, as specific words often have an associated legal meaning.[162] Coupled with the continued use of arcane Latin words and phrases, understanding the legal language requires regular visits to a specialized reference text: Black's Law Dictionary. Importantly, what Mellinkoff points to is an existent conversion process between legal and ordinary language. Despite the language of the law being housed within the same linguistic vehicle (i.e., natural language) as common speech, the differences in the lexicon are sufficiently vast such that translation is required.

However, Mellinkoff's discussion around the seemingly esoteric vocabulary of the law serves the purpose of an incisive commentary. He notes that the historical reasons for their existence are often justified as reasons for their current use. The bridge between the vocabulary and arguments for their continued practice center around the discussion on precision. Mellinkoff underlines the law's play on, and perhaps obsession with, being precise. He considers the law's characterization as both one of "extraordinary precision" and full of "weasel words."[163] Precision is a deliberate choice, whereby flexibility is deployed intentionally. In effect, Mellinkoff delineates the boundaries, in the interpretative space, between clarity and precision.

He describes the legal language as a "viscous sea of verbiage,"[164] leading to a 'muddiness' in understanding. For Mellinkoff, "if there is any meaning, it is hard to find."[165] Interestingly, he defines this lack of clarity on the basis of several structural peculiarities: (1) long sentences;[166] (2) awkward

[162] *Id.* at 12.

[163] Mellinkoff cites H. Cairns in "Language of Jurisprudence" 232, 259 (1957) and Stuart Chase, *The Tyranny of Words* 324 (1938). See *id.* at 21.

[164] Mellinkoff cites Stuart Chase, *The Tyranny of Words* 327 (1938). See *id.* at 24.

[165] *Id.* at 25.

[166] Mellinkoff suggests an average of one hundred seventy-six words in a sentence is not anything out of the ordinary. *Id.* at 26

sentences;[167] and (3) "tortured metaphors."[168] These three factors contribute to the inability and failure of the language to communicate. Clarity is then associated with the capacity to communicate, whereas precision is related to the choice of vocabulary. As a result, clarity is directly correlative with meaning-making. Precision is merely stylistic, an aesthetic decision.

So why then is Mellinkoff concerned with precision? He argues that precision is often referenced as the virtue and response to any criticism against the language. The characteristic of being precise is fundamental to its existence. Moreover, precision fosters accuracy; the language of the law is exact. Consequently, even if the language is obscure, it is necessarily so.[169] Evidently, Mellinkoff alludes to the irony of the legal language; that form precedes substance. Precision is intimately linked with certainty and the ancestry of its use. For these reasons, precision reigns over clarity. Other defects of the language are a small sacrifice in exchange for precision.

While Mellinkoff interprets clarity as a substantive trait, he concedes that the argument for precision often traverses into the territory of clarity. That is, those familiar with the language consider that precision enables clarity because the vocabulary is already understood.[170] This suggests that Mellinkoff is posing an argument not to the legal community, but more broadly, to the public with the subtext of breaking down the barrier to legal literacy.

Oddly, he brings to light two variations of precision: (1) exact; and (2) "exactly-the-same-way." The former implies well-defined limits. The latter is an appeal towards tradition and the tool of precedent, laced with 'magic words' and birthed from religious ritual.[171] Mellinkoff's subsequent discussion of the two, interestingly, raises the issue around their interchangeability. He suggests that the two kinds of precision are often treated as if there is no distinction. Precision is often conflated with tradition because sufficient repetition of the ritual words produces the effect of being exact. As a result, precise language applies the strengths of the first variant, but, in fact, justifies the practice of the second. Furthermore, he argues that the meaning is indifferent, "for all language is arbitrary."[172] Again,

---

[167] Mellinkoff suggests even shorter sentences, with innumerable dependent clauses make sentences unclear. See *id.*

[168] Here, Mellinkoff points to imagery and artful use of literary devices that do not provide any useful information. See *id.*

[169] Mellinkoff considers the argument raised by Sir Ernest Gowers on legal language. See *id.* at 291.

[170] *Id.* at 292.

[171] *Id.* at 296.

[172] *Id.* at 299.

Mellinkoff refers to the distinction between clarity and precision. For it does not matter whether the language is clear, what matters is that its practice is upheld. Mellinkoff, therefore, regards the legal language as divorced from its substance; it is a product of mindless linguistic formulae.

Perhaps again with an ironic touch, he reflects on definite meaning and whether the language of the law has ever had any. He concludes that the "only reason for [the language's] existence" is what he labels as "flexibles."[173] Mellinkoff points to the classic example of the word: *reasonable*. There has never been a 'real definition' around the term.[174] Yet, legal text is riddled with this word. What Mellinkoff suggests is evident; the arguments for precision, often grounded in certainty, bury the language's heavy dependence on flexible words. While reasonable is an obvious instance, he considers other words that are not observably vague. He references Old and Middle English words, such as *aforesaid*, *heretofore*, *forthwith*, and *hereafter*. The most infamous is *whereas*, the "most persistently typical and most consistently vague words in the language of the law."[175] Mellinkoff notes that the word takes on innumerable meanings, often with immense polarity. *Whereas* became a term of art when English legal forms were hardened in the eighteenth century, borrowed from the "loose usage" in Middle English common speech.[176]

Many of the Old and Middle English words used in legal language were taken from common speech. While their meanings have changed, their spelling has not. This pattern of borrowing, coupled with an insistence on tradition and repetition of practice, has subverted a cognitive recognition of change. That is, the changes in meaning were effectively a translation process that has been forgotten with time. Words that may have once been precise, have lost their cut and "sucked dry of reason."[177] This is reminiscent of Stanley Fish on the use of canonical texts. Fish regards the significance of language as characterized solely by the "realm of value and intention but begins and ends with that realm."[178] Language carries obligations and commitments that were once undertaken but eventually assumed; thereby rendering inseparable its original intentions at its core.[179] As a result, inherent philosophical

[173] *Id.* at 301.

[174] Mellinkoff cites Chief Justice Goddard in his opinion in ***R. v. Summers***, [1952] 1 All E.R. 1059, 1060. See *id.* at 303.

[175] *Id.* at 321.

[176] *Id.* at 323.

[177] *Id.* at 326.

[178] Stanley Fish, *Is There a Text in This Class? The Authority of Interpretative Communities* 107 (1980).

[179] *Id.* at 108.

and moral concepts are 'built into'[180] the language such that over time its interpretative exercise is forgotten and accepted as fact.

The problem is that canonical materials "carry their authority…seeming to have acquired it by natural right…not to encourage thought but to stop it."[181] For example, the process of making language 'ordinary' allows for the repurposing of words and grammar without the need to reintroduce the politics. Therefore, the illusion of language being transcendent, logical, and independent of meaning is merely a product of perverse procedure. This suggests that at the core, the mechanism of ordinary language that builds abstraction and principle is able to invent and reconstruct without truly breaking from its original form. Linguistic practices that have emerged through sociopolitical contexts are understood as the legitimate language with its normativity buried deep within its practice.

Mellinkoff's discussion on the persistent use of Old and Middle English in legal language reflects a disjunct relationship between concept and structure. His solution is to then discard the complexities (peculiarities) of the language and use, in its place, everyday common speech. Though the intention of Mellinkoff is to argue that these antiquated and archaic practices should be removed, his argument, in fact, points to a deeper problem of translation. If "legal terms of art" were borrowed from once common speech, would removing these practices, in the name of aligning with current plain language, not reinforce the exact problem Mellinkoff is hoping to resolve? That is, how often must a realignment process occur in order to ensure that legal language is sufficiently communicative and consistent with ordinary language? What are the temporal limits to common speech?

There are parallels found in Giorgio Agamben's work and his regard of language as a reliquary signature to an analogical and immaterial model.[182] Signatures operate as archaeological traces that represent how nondescript objects connect to events and/or subjects. Signatures characterize and specify, while signs provide its conditions.[183] Though the signature itself is void of content, they enable the efficacious existence of the sign. Without the signature, the concept will remain inert.[184] Is the legal language, the juridical formula, an artifact of another time? Or, is the legal language a

[180] *Id.* at 107.

[181] Stanley Fish, *The Trouble with Principle* 47 (1999).

[182] Giorgio Agamben, *The Signature of All Things* 36 (2009).

[183] *Id.* at 79-80.

[184] *Id.* at 76.

transcendental signature? Therefore, Mellinkoff questions the necessity of having a unique and distinct legal language. More importantly, he raises the argument of whether the language is sufficiently serving its role to convey legal knowledge.

In unpacking Mellinkoff, it is unavoidably evocative of George Orwell and his distaste for written English in political texts. In 1946, Orwell wrote in his essay "Politics and the English Language:"

> ...the abuse of language is a sentimental archaism, like preferring candles to electric light or hansom cabs to aeroplanes. Under this lies the half-conscious belief that language is a natural growth and not an instrument which we shape for our purposes.[185]

Orwell discusses the "bad habits" of writing that are spread by "imitation."[186] The lack of precision characterizes English prose, marked by vagueness and indifference to word choice. That is, words are not chosen for their meaning, but "phrases [are] tacked together like the sections of a prefabricated hen-house."[187] Perhaps with the same indignance, Orwell points to the "worn-out metaphors which have lost all evocative power,"[188] complex verb constructions and use of the passive voice, and pretentious diction that "give an air of scientific impartiality to biased judgments."[189] It is exactly these qualities that, Mellinkoff suggests, color and corrupt the legal language. Yet, these traits described by Orwell are found in political writing. Like Mellinkoff, Orwell argues that words that have outworn their usefulness should be discarded. As well, all "prefabricated phrases, needless repetitions"[190] should be cut. But, where Orwell and Mellinkoff diverge is their respective views on simplification of language. While Mellinkoff merely alludes to simplifying the language, Orwell tackles simplification and its relationship with meaning.

To Mellinkoff, simplifying the vocabulary and syntax appears to be a quick fix for the murkiness of the legal language. To reiterate, he argues that legal language should align with ordinary language. Orwell interestingly ventures further. Specifically, he highlights the notion of "fake simplicity and the

[185] George Orwell, "Politics and the English Language," in *Why I Write* 102 (2004)

[186] *Id.* at 103.

[187] *Id.* at 105.

[188] *Id.* at 106.

[189] *Id.* at 107.

[190] *Id.* at 119.

attempt to make written English colloquial."[191] As opposed to "setting up a 'standard English' which must never be departed from,"[192] Orwell focuses on concreteness and meaning first. He describes this as a conscious effort to predicate meaning over word choice. Though Mellinkoff and Orwell both argue that language expresses thought, Orwell raises the question of how thought can dictate language. Inadvertently, he reaffirms that form and substance are indeed distinct. However, clarity and precision are one and the same: they define meaning.

This may be revealed through the six rules Orwell believes would enable better communication of thought:

> (1) Never use a metaphor, simile or other figure of speech which you are used to seeing in print.
> (2) Never use a long word where a short one will do
> (3) If it is possible to cut a word out, always cut it out.
> (4) Never use the passive where you can use the active.
> (5) Never use a foreign phrase, a scientific word or a jargon word if you can think of an everyday English equivalent.
> (6) Break any of these rules sooner than say anything outright barbarous.[193]

Perhaps an exact reflection of Mellinkoff's arguments, traces of Orwell's rules have equally been found in descendants[194] of Mellinkoff's work. Peter Tiersma, in *Legal Language*, reflects on how well the language of the law operates as a means of communication. Tiersma suggests that other uses and goals, including the "desire to appear objective and authoritative" and the use of the language as "a marker of prestige and badge of membership," take precedence over communicability.[195] Tiersma then answers Mellinkoff's question: the legal language does not serve a communicative function.

Tiersma's text acts as a counterpart to Mellinkoff's work. Similar to Mellinkoff, he begins with a walk through the ancestry of the legal language. However, Tiersma considers the retention of Latin and other legal archaisms as the consequence of natural evolution. For Tiersma, the legal language is representative not only of the influence of diverse culture, but also a reflection of the growing

---

[191] *Id.* at 118.

[192] *Id.*

[193] *Id.* at 119.

[194] Richard C. Wydick wrote an entire book titled, *Plain English for Lawyers*, that is an exact mirror to the Orwell's rules. *See* Appendix A for how chapter titles are a reformulation of Orwell's rules.

[195] Peter M. Tiersma, *Legal Language* (1999), available at: http://languageandlaw.org/LEGALLANG/LEGALLANG.HTM.

complexity of the legal system.[196] Nevertheless, Tiersma concludes that the legal language has enabled lawyers to retain monopoly on the provision of legal services and, in effect, maintain the legal fraternity.[197]

Like Mellinkoff, Tiersma raises parallel arguments on the strategic use of precision as well as the unique legal lexicon that is representative of the language. Tiersma, though, extends Mellinkoff's observations. Building on Mellinkoff's discussion of "uncommon meanings,"[198] Tiersma highlights the frequent application of "legal homonyms."[199] That is, legal terms either wear two or more meanings; or that they have a divergent legal from ordinary meaning. As well, Tiersma discusses other markedly legal traits. First, legal sentences appear to pivot around modal verbs like *shall*. Though *shall* tends to serve a temporal function in ordinary language, in legal language, *shall* frequently signals obligation. Moreover, legal language is significantly dependent on reference. Tiersma notes the linguistic difference between referential and attributive descriptions. Attributive refers to a general entity that fulfils a particular description, whereas referential denotes a specific entity.[200] Certain legal texts (e.g., wills, contracts, etc.) intentionally play on referential and attributive descriptions. Legislative documents, however, almost always use attributive references. This, in turn, enables multiple interpretations and referential ambiguity.

While Mellinkoff is largely concerned with lexical complexity, Tiersma alludes to Mellinkoff's note on structural and syntactic complexities of legal language. In particular, he continually refers to the "unusual sentence structures" of the language, including conjoined phrases, impersonal constructions, "an inordinate amount of negation," and the "separating of subject from the verb, or splitting the verb complex."[201] These 'quirks,' as Mellinkoff suggested, do not have any communicative purpose, and could easily be removed. Yet, Tiersma argues that these stylistic features reveal that the legal language follows its own set of linguistic rules. That is, the distinct 'characteristics' of the legal language are, in fact, inherent to its formulation. The legal language contains syntactic and semantic constraints, along with a unique grammar. The language of the law

[196] *Id.*

[197] *Id.*

[198] Recall Mellinkoff's discussion on common words with uncommon meanings.

[199] *Id.*

[200] *Id.*

[201] *Id.*

is its own separate language. In short, Tiersma's analysis brings to light how unpacking linguistic constructions not only muddies the divide between form and substance, but may also be crucial to understanding the language's unique code.

Law's Language

In the aforementioned section, Mellinkoff chronicled how the language of the law is a product of historical legacy and tradition, explanatory of its archaisms and structural form. That is, the language is a mere consequence of ritual and ecological inheritance. He notes that its specific and unique characteristics are matters of form and not substance. This means that law and its language are suggestively distinguishable. Conceivably, then, the language is not married to the discipline and transforming legal to plain language is possible. This suggests that legal concepts are capable of being extracted from their current arrangement and transposed into an ordinary, everyday linguistic form. The technical language is simply embellishment. This view, however, is not shared by Tiersma. Rather, he raises the argument that the relationship is less distinct; that the language is, in fact, integral to its function. The question becomes: does the performance of the legal language affect its existence?

In *Legal Discourse*, Peter Goodrich provides a careful account on the perceptions of language in legal contexts. In particular, Goodrich highlights the science of legal language, placing emphasis on its regard as an independent, technical language as opposed to a specific category embedded within an "existent language system."[202] In prior literature, language has been described as a neutral instrument used to justify the application of formalistic legal methods. Goodrich provides a critique of this notion, putting forth arguments for the social and political dimensions of legal semantics. Interestingly, Goodrich alludes to linguistics and jurisprudence as parallel operations, both relying on the 'codes' that govern. Across both disciplines, the attention has largely dwelled on the "abstract imperatives," captured as an objective study without regard for its subjective context.[203] Instead, Goodrich argues that to understand linguistic and semantic inclusion is, in effect, to bring to light the relationship of law and power. Law as a genre of linguistics tackles meaning at its heart. Therefore, Goodrich appeals to the interrelations of form and substance, and the privileging of structuralist over historical account.

---

[202] Peter Goodrich, *Legal Discourse: Studies in Linguistics, Rhetoric and Legal Analysis* 2 (1985).

[203] *Id.*

Goodrich raises a fascinating objection to structural linguistics in law. Structural linguistics perceives language as a medium reducible to scientific form, creating the illusion of conceptual universality. In the same manner, the use of language in legal practice implies consistent rules of internal governance according to a static, positivist grammar. In other words, Goodrich considers that the dominant paradigm of language analysis is a justification for legal formalism and treatment of text as "predicated upon its unity as the expression of a precedent intention or will."[204] This suggests that the process of determining meaning has largely followed an "analytic reconstruction of its source."[205]

To paint this picture, Goodrich turns to the jurisprudential work of Hans Kelsen and the "pure science of law."[206] Kelsen's logical analysis of law reduces the chaos of perception to a "multitude of general and individual norms."[207] These norms fulfil logical conditions for an objective interpretation of law.[208] Like Kelsen's norms, Ferdinand de Saussure's conception of linguistics rests on the principles of formal validity and order. Saussure's articulation of a general linguistics draws attention to the semiotics of legal argument.

Saussure, perceived as a close ancestor to modern linguistics, regarded language as a system: an institution of the present, but also a product of the past.[209] In the nomenclature developed by Saussure, the words of a language are understood as a "two-sided psychological entity"[210] – the signified (concept) and the signifier (sound pattern). The former builds the connection to the latter and is institutionalized in the language. Therefore, the linguistic sign is considered whole when both constituent elements are present. The connection between both elements is arbitrary; there is no internal association between the signified and the signifier.[211]

As there is a disconnect between the signified and the signifier, Saussure's linguistic system is predicated on a method of reference and classification. Meaning is not anchored in reality but only understood through conceptual relations. Meaning is determined by the relational contrasts of words

[204] *Id.* at 3.

[205] *Id.*

[206] Hans Kelsen, *Pure Theory of Law* (first published in 1934, Max Knight trans., 1967).

[207] *Id.* at 72.

[208] *Id.* See also Goodrich, *supra* 202 at 38.

[209] Ferdinand de Saussure, *Course in General Linguistics* 10 (Bloomsbury Revelations ed. 2013).

[210] *Id.* at 77.

[211] *Id.* at 78.

within the system, otherwise perceived as *associative* representations of reality. Similarly, the grammar of legal language reflects the "scientificity of the normative order and the necessary interrelation of its elements, the status of the system itself as a series of necessary (analytic) entailments."[212]

Evidently, this is reminiscent of systems theory. Systems theory conceives law as a social system that generates its own realities and languages with processes and modes of classification.[213] As in structural linguistics, the law is ordered, consistent, and internally coherent. Niklas Luhmann understood law as 'semantic closure' such that its high degree of internal complexity, self-reference, and self-modification is indicative of how the law evolves.[214] The law is a "structure of *symbolically* generalized expectations;"[215] with no concrete fixed definition but a "surplus of references."[216] The legal system draws its boundaries through language. Jürgen Habermas, on the other hand, viewed the law as a mode of interconnectivity – "an integrating factor that links the lifeworld to these systems."[217] Habermas suggests that law is itself a translator, allowing different spheres to communicate meaningfully. Law institutionalizes the rational will of the lifeworld through language and is amoral.[218] The law is made objective through its language.

While their views of the legal 'system' diverge, both Habermas and Luhmann could agree that the purpose of language is to perform "... to a high degree of accuracy and transparency, the task which law sets for it,"[219] reflecting an impartial distance between law and language. Language then is distinct from the law, functioning merely as law's surrogate for stability and predictability. This suggests that the demands of the legal language are relatively simple: language must operate independent of

---

[212] Goodrich, *supra* 202 at 39.

[213] Chris Hutton, *Language, Meaning, and the Law* 24 (2009).

[214] *Id.* at 25.

[215] Niklas Luhmann, *Law as a Social System* 146 (2004).

[216] *Id.*, at 144.

[217] The 'lifeworld' is defined as the general 'private' sphere; the everyday world of family, social practice and beliefs that sustain the 'public' sphere. They form the horizon for speech and the source of interpretation and is reproduced only through ongoing communication. *See* Jürgen Habermas, *Between Facts and Norms: Contributions to a Discourse Theory of Law and Democracy* 354 (1996).

[218] Chris Hutton, *supra* 213 at 28.

[219] *Id.*, at 29.

meaning. Returning to Saussure, an analogy appears whereby the legal language may be seen as the signifier, while legal substance is the signified.

Duncan Kennedy pioneers the structural linguistic analogy to legal argument by deconstructing the language as a system of 'argument-bites.' Argument-bites form the basic unit. Operations then performed on argument-bites constitute and build legal arguments. Such operations diagnose and assume the circumstances, or relationships, in which the argument-bite is to be manipulated and 'deployed.'[220] Such import of structural linguistics conceptualizes law and argument as systematically formulaic; "a product of the logic of operations."[221] Perhaps most interesting about Kennedy's theory is his idea of 'nesting.' Kennedy describes nesting as the act of 'reproduction' or the "reappearance of [argument-bites] when we have to resolve gaps, conflicts or ambiguities that emerge [from]…our initial solution to the doctrinal problem."[222] In reality, nesting arrives when courts are asked to rule on inherently subjective standards of reasonableness.[223] Therefore, the conundrum surfaces where language may be applied to law in a mechanical fashion but the process of reducing legal argument to a system of operations raises considerations on the act of labelling and the power in its performativity. That is – and as Kennedy rightfully notes – "language seems to be 'speaking the subject,' rather than the reverse."[224]

Within the concept of meaning, there is then an objective and subjective variant. The objective legal meaning represents the product of the "presupposition of the basic norm as the principle of origin and the criterion of validity for legal norms."[225] Referring to Kelsen, these norms intend only to describe the system, but its actual practice is considered unimportant. Description is "an abstraction away from a social practice embedded in the multidimensional normativity of the social world."[226]

---

[220] Kennedy describes relating argument-bites to one another by such operations as a means of confronting legal problems. *See* Duncan Kennedy, *A Semiotics of Legal Argument*, 3 Collected Courses of the Academy of European Law 317, 351 (1994).

[221] *Id.* at 343.

[222] *Id.* at 346.

[223] Kennedy suggests nesting arises out of its association with objectivity; judges "prefer it because it harmonizes with the stereotypically judicial pole in the judge/legislator dichotomy." See *id.* at 348.

[224] *Id.* at 350.

[225] Goodrich, *supra* 202 at 40.

[226] *Id.*, at 34.

This enables the distancing between the substantive production and scientific maneuvering of legal norms.

This separation between form and substance fosters an "agnostic semantic subjectivism," such that it is "futile or fictitious even to attempt to specify any single, correct, interpretation or application of a general norm."[227] He then highlights other legal philosophers, in comparison to Kelsen, in an attempt to reinforce the formalist paradigm of language analysis. Goodrich cites H.L.A. Hart, for example. He argues that Hart's rule of recognition is a mere reformulation of a formalistic analysis of law, but as a mutually reinforcing system of rules. Hart's contribution is only a minor revision to a fundamentally structural account on legal validity.[228]

In this regard, Goodrich bridges Hart with Ludwig Wittgenstein. Accordingly, the actions derived from the word are effectively married to its meaning. Language is a form of life.[229] Linguistic expression is, therefore, constructive of its being. It is conceivable then that language could be no more than a list of orders and classifications. It follows that in abiding by the rules of association – or, to play the game – is to accept the inherent authority of its practice. Meaning is found in the performance of the word, and not in the understanding of it. The 'language-game' clarifies the context which binds its use and, in effect, its meaning. What Goodrich emphasizes is that there remains a distinction between the internal character of the law,[230] and the external usage.

The problem with Goodrich's argument is that he conflates traditional linguistics with the philosophy of language. In particular, he defines language study as one limited to objective idealism and the ghost of semiotics; that regard for language in law has continually focused on rough reconstructions of Saussure's principles. He sees that the dominant framework disregards the politics of legal interpretation and focuses on asserting logic to legal language. In effect, the law is scientifically captured within a structural grid of analytical conditions and constraints. So, what is Goodrich's response to the "evil hand of formalism"[231]?

---

[227] *Id.* at 43.

[228] *Id.* at 48.

[229] Ludwig Wittgenstein, *Philosophical Investigations* 19 (2nd ed. 1958).

[230] The consideration that the character of the law is a social fact. Goodrich suggests this is merely 'descriptive sociology' of legal substance. *See* Goodrich, *supra* 202 at 48.

[231] *Id.* at 55.

A decisive turn from logic, Goodrich, therefore, proposes the integration of sociolinguistics to legal analysis. He argues that the role of linguistics should account for law as social action.[232] That is, it should consider the inequalities of power that are syntactically embedded within the system. Texts are a "complex combination of linguistic constructions, functions and codes correlated to variable socio-political and ideological contexts."[233] As opposed to linguistic structure, the focus should be on linguistic effects, and specifically the effects of discursive processes. Consequently, Goodrich suggests that rhetoric should instead be the focus of language study, encapsulating the existence of "legal fictions and legalistic abstractions"[234] and logical fallacies inherent in legal text. Moreover, rhetoric studies the forms of discourse, particularly those of literary genres including metaphor and metonymy.[235]

This particular strain of understanding draws flavors of Lon Fuller and his essays on legal fictions. Fuller describes the status of legal fictions as linguistic phenomenon. More importantly, legal fictions vis-à-vis legal and scientific facts were of particular concern to him. Fuller considered legal fictions as a litmus test on the boundaries of the language. Defined as "consciously counterfactual propositions,"[236] he referenced legal fictions as a specialized form of linguistic abstraction. Fictions have the constructive function to "keep the form of the law persuasive."[237] In effect, legal fictions are rhetorical devices, representative of the linguistic mechanisms that enable legal processes.

Similarly, Goodrich suggests that legal language must turn to its communicative function and its capacity as the discourse of power.[238] In contrast with the "determinate logic of legal signification,"[239] often framed as instruction, rhetoric stresses argumentation. Rhetoric is concerned with the use of language to enable a given result. Though Goodrich focuses on the significance of speech, he does not perceive it in the same light as J.L. Austin. For Goodrich, Austin's reflections on speech acts and performativity remain in the realm of structure. Again, Goodrich regards Austin as a distant ancestor

[232] *Id.* at 76.

[233] *Id.* at 79.

[234] *Id.* at 86.

[235] *Id.* at 87.

[236] Lon Fuller, *Legal Fictions*, 25 ILLINOIS L. REV. 363, 369 (1930a).

[237] *Id.* at 387.

[238] Goodrich *supra* 202 at 88.

[239] *Id.* at 88.

to Saussure and semiotics. Austin's speech acts describe how legal obligations are relative to public specification; utterances necessarily correspond to particular procedures situated within social contexts. Their mis-performance leads to a nullification or voidance of the act.[240] Utterances are akin to directives for 'appropriate' social behavior. Language has a definite sense and reference. For words to have meaning, their reporting must necessarily ascribe to these attributes.

In contrast, Goodrich aligns with the arguments of Stanley Fish on text and the role of the audience. Fish draws the connection between assumptions and argumentation. He suggests that questions formed against the linguistic problems are mere projections of the readers themselves. As a result, the interpretation of arguments changes with the reader such that meaning reflects not the capacity of expression, "but the ability of a reader to confer it."[241] Therefore, it is naturally contradictory to conceive of language as neutral constructs. The consideration of language as one that simply mirrors facts independent of purpose or perspective, is a fiction.[242] Perhaps, as Michel Foucault states, rather than 'an arbitrary system,' language forms are interwoven with the world. They are an "enigma renewed in every interval…and offer[ed]…as things to be deciphered."[243]

How language constructs reality is an important question. Goodrich suggests that, against determinacy, rhetoric focuses on persuasion as a relative concept and is subject to probability.[244] The content of a word is both conventional and temporal, storing references of the time. It is a normative scheme that does not offer formal proof but is indicative of the context and power that underpins and guarantees its authority. Goodrich then is preoccupied with institutional determination of meaning: to develop an understanding of the "frequently obscured persuasive, argumentative and coercive levels inherent in the writing of legal texts."[245]

He considers that the use of linguistic mechanisms enables the law to appear as if there is a consensus on social values and justice. The legitimacy of the law is presumed but requires explaining.[246] Goodrich, therefore, reconceptualizes from 'how to do things with legal words' to 'how do legal

---

[240] John L. Austin, *How to Do Things with Words* 16 (2nd ed. 1975).

[241] *Id.* at 9.

[242] Fish, *supra* 178 at 106.

[243] Michel Foucault, *The Order of Things: An Archaeology of the Human Sciences* 35 (1970).

[244] Goodrich *supra* 202 at 102.

[245] *Id.* at 97.

[246] *Id.* at 117.

words do things.' His argument reflects on the manipulation of linguistic practices that produce divergent meanings. As opposed to denoting a legal meaning, Goodrich points to the way in which meaning that already exists in a particular social context is intentionally used in a legal environment. Importantly, meaning must be understood as a consequence of institutional appropriation, its discursive formation as a network understanding of both the internal ordering and relationship to other discourses.[247] In short, language becomes a social phenomenon whereby form and substance are inseparable.

For linguists, there is a distinction between philosophy from the practice of core linguistics. While Goodrich has identified structural linguistics as having removed semantics, and thereby diluting methods of realizing meaning, his proposal equally diminishes semantics and other linguistic practices of deriving meaning. This is because Goodrich mistakenly construes syntax as interchangeable with semiotics and semantics; in effect, conflating several linguistic fields under a single umbrella. His proposition is conceivably a pairing exercise between legal and linguistic theory, rather than a substantive legal analysis through a linguistic lens. Goodrich's text appears then to juxtapose somewhat antiquated notions of structural linguistics against highly contextualized discourse analysis. This produces a sharper distinction and builds a stronger justification for his argument but fails to accurately capture the role of linguistics in law. Nevertheless, where Goodrich succeeds is precisely the consideration of discourse and context as essential to language study.

In an earlier account to Goodrich, Brenda Danet provides a thorough linguistic analysis of the interrelations of law and language. She pioneers research on the use of language to perform law's core functions. She describes these functions as (1) the ordering of human relations; and (2) restoration of social order.[248] Importantly, Danet offers an initial framework for the study of language in law, with specific concerns from the perspective of sociolinguistics. Goodrich and Danet appear to be two sides of the same coin. Goodrich, however, is a legal scholar; Danet is a communications and sociolinguistics expert. It follows that her contribution delivers a necessary counterbalance to the aforementioned discussion. It must be noted that Danet's arguments extend beyond written texts and into the realm of dispute and trial analysis. For this reason, sociolinguistics equally factors

[247] Goodrich coins the terms intradiscourse and interdiscourse to describe the system of discursive formations. See *id.* at 144-151.

[248] Danet, *supra* 157 at 449.

behaviors of individuals within the courtroom setting. These considerations fall outside the scope of this thesis.

In her text, Danet begins with an introduction to the notions of competence and performance drawn from renowned linguist, Noam Chomsky. Chomsky separates the capacity to produce with the actual use. For Chomsky, linguistic knowledge is independent of its environment. Chomsky's model obsesses over a strict adherence to systems theory. That is, language is an entirely internal system, with inherited forms of organization that are agnostic to features of the environment.[249] As linguistics is divorced from its speakers and societal embedding, Chomsky's language system is outside of evolution. Its rules remain constant in spite of external changes. Danet considers Chomsky's theory as the separation of internal language rules from outward engagement.

Like Goodrich, she raises hesitations around this perception and argues for the consideration of context in deriving meaning. In comparison with Goodrich's logical divide, Danet draws a distinction between semantics and pragmatics on the premise of context. The next chapter will dive deeper into these linguistic fields. But, for the intentions of clarifying her argument, semantics alludes to sentence meaning that is context-independent and pragmatics is context-dependent, drawn entirely from interpretative acts. As will be seen, Goodrich fails to detail the various ways in which pragmatics manifests itself in legal language. His discussion on rhetoric only articulates one area of pragmatics: discourse. Danet, on the other hand, captures holistically the variable field of pragmatics as the layer on which the functions of the law are revealed.

Danet argues that the significance of pragmatics is particularly noticeable when distinguishing between meaning as an object and meaning as an act.[250] Meaning as an object returns to discussions of objectivity and "correct" characterizations of reality. Meaning as an act, on the other hand, is constructivist and a result of knowledge that extends beyond the information given in a particular text.[251] The dichotomy is further accentuated when reflecting on literal as opposed to metaphorical uses of language. In the constructivist perspective, metaphor is not a form of embellishment, but a feature of the game. Contrary to Goodrich, Danet finds that Wittgenstein's language games view language precisely in context. To Danet, the referential correspondence between the word and use

[249] Hutton, *supra* 213 at 38.

[250] Danet, *supra* 157 at 455.

[251] *Id.*

can be regarded as tools in a toolkit.[252] In contrast, Goodrich's interpretation of Wittgenstein conceives of legal language as a single closed system. Alternatively, Danet considers legal language as a simultaneous engagement of multiple language games, deployed and played differently in accordance with circumstance.

Moreover, meaning as an act highlights a difference between sentences and their empirical use. Though Goodrich alludes to performativity, his rejection of Austin demonstrates a regard for utterances from a one-dimensional lens. To Goodrich, Austin's performatives are mere instructions. Danet, instead, suggests that Austin's work captures the institutional authority of the law and, in fact, are the foundation on which legal relationships have come to be expressed. Scholars like John Searle later developed typologies[253] that build from Austinian performatives. Several categories of speech acts are of particular importance: (1) representatives; (2) directives; (3) commissives; and (4) declarations. Representatives are utterances that assert the truth of propositions. They set the reality in which the utterance occurs. Commissives are utterances that behave as future commitments.[254] Danet draws the analogy between commissives with promises and contracts.

Directives and declarations are perhaps the most intuitive connection. Directives are utterances found largely in legislative documents and considered, by default, obligations. Directives are a marriage of form and substance as the context of its use is implicit of its authority. Declarations are utterances that, when successfully performed, "bring about a correspondence between their propositional content and reality."[255] That is, there is a change in state predicated upon both linguistic competence and the extra-linguistic institutional authority of the speaker. Within Searle's categories, the subgroup of representative declarations is striking. Coupled with the notion of representatives, Danet points to the mythical fact-law divide. The successful performance of legal utterances does not require the ascertainment of facts. Instead, they define what are the facts; and thereby assert a legal reality. This parallels Geoffrey Samuel's discussion of legal reasoning as the manipulation or

[252] Danet, *supra* 157 at 456.

[253] John Searle, *A Classification of Illocutionary Acts*, 5 Language in Society 1 (1976).

[254] Danet, *supra* 157 at 459.

[255] *Id.*

construction of 'virtual' against perceived 'actual' factual situations.[256] Facts of a case do not exist until they are constructed through argument.

In short, Danet reveals that speech acts are a pragmatic dimension that express the law's institutional power and construct binding relationships between parties. Interestingly, Danet reflects on discourse analysis. She describes this development as a subfield of pragmatics concerned with "how the parts are linked to the whole."[257] This means that discourse describes the cohesion of the language or the coherence of a series of utterances; in other words, fractals. Discourse analysis serves as a test of interoperability and consistency across the legal system, expressed through the language. Counter to Goodrich's argument, discourse analysis alone insufficiently articulates the dynamics of power embedded within the language. Alternatively, Goodrich provides a strong basis of how rhetoric enables constructions of truth; perhaps suggesting a misinterpretation of discourse analysis as interchangeable with rhetoric.

After laying the theoretical foundation, Danet considers the linguistic status of legal language. Is legal language a technical dialect? She considers legal language as a form of diglossia – a variant of higher prestige "superposed" on to the native practice.[258] Interestingly, she notes that the complexities of legal language are the complexities of natural language. In effect, "the indeterminacy of the law is in part the indeterminacy of the language itself."[259] In this manner, the attempts at clarifying legal language stipulated by Mellinkoff are rather futile. Evidently, shifts away from specific legal jargon would not have a substantive impact on the clarification of legal meaning.

Danet points to an exploratory study[260] on the conceptual and linguistic complexity of legal language. She explains that despite linguistic reform,[261] comprehension did not improve. Moreover, she unpacks the argument frequented by the legal community that "legal concepts are inherently difficult and cannot be simplified." In another study,[262] she observed that, contrary to the perceived outcome, greater conceptual difficulty did not lead to reduced comprehension. There is then a gap between

[256] Geoffrey Samuel, *Is legal reasoning like medical reasoning?*, 35 LEGAL STUDIES 323 (2014).

[257] Danet, *supra* 157 at 463.

[258] *Id.* at 473.

[259] *Id.*

[260] *Id.* at 488.

[261] By linguistic reform, Danet specifies syntactic and lexical simplification. See *id.*

[262] *Id.*

comprehension and simplification. Danet argues that this is because linguists do not treat clarity and simplicity as equivalents; "language has important functions beyond referential."[263] She states that "no amount of simplification of language…can guarantee that its [legal] conditions are fair. Fairness is a substantive issue, not just a formal one."[264] As a result, the issues of clarifying legal language are not easily resolved through syntactic or semantic simplification. Instead, there must be consideration of both substance and form. The subsequent case studies in the following chapter will further explore the distinction between clarification and simplification.

The central proposition of Danet's text is her discussion on the "thickening" of legal language. She argues that while law appears to deal with fact, the preoccupation with a highly elaborate and esoteric language suggests that the function is not referential, but poetic.[265] Recalling Fuller, the active use of legal fictions, a consciousness of falsity, is both a distinctive and embedded function of the language. Written legal documents perform in a manner opposite of its claims towards precision, transparency, and truth. Text is a source of symbolic significance, birthed from ritual and bears the aesthetics of mystery. It is akin to religious discourse, sufficiently cryptic to be unquestionably true.[266] Though Danet offers several explanations, perhaps the most convincing is that legal language maintains an illusion of certainty amidst "a world of uncertainty."[267] Perhaps it is as Orwell suggested, the legal language is designed to "give an appearance of solidity to pure wind."[268] The language is meant to be experienced and not understood.

Evidently, several crucial lessons may be drawn from Danet. First, applying the philosophy of language as a lens of legal analysis produces distinct discussions from core linguistic analysis. Second, to undergo a linguistic analysis of legal texts is to necessarily consider pragmatics. Accepting the premise that legal language is constructivist, ambiguity is then inherent to language. As well, conceptual complexity does not need to be reduced to increase comprehension. Simplification does not necessarily lead to clarification. Equally, rhetoric plays a critical role in legal language, such that it reveals the mechanics of legal reasoning and the myth of "fact-finding." Danet points to the

[263] *Id.* at 490.

[264] *Id.* at 489.

[265] *Id.* at 540.

[266] *Id.* at 545.

[267] *Id.*

[268] Orwell, *supra* 185 at 120.

poeticization of legal language. Intrinsic to the language is the veil of mystery reinforced by literary device. As opposed to the language used in legal contexts, what Danet reveals is the instrument of language in legal processes. All in all, Danet's text may be perceived as a response to Mellinkoff and, arguably, provides a better account of how the law is a medium of communication. Her lessons are located within an existing body of legal theory, perhaps indicative of core linguistic analysis as an effective frame of legal analysis.

Law as Language

Thus far, the chapter has explored the intricacies of legal language, reflecting on the uniqueness of its personality. Danet questioned the linguistic status of legal language, whether it is a technical dialect or a variant. More importantly, it has been regarded that embedded in the language are the dynamics of institutional power. That is, legal language wears a cloak of authority that can be distinguished from ordinary language. The following section builds on this notion to uncover perceptions of law as a linguistic vessel.

James Boyd White instructs his readers of the contours of legal language and the lawyer as a writer in *The Legal Imagination*. This work has been renowned to introduce how identities and meanings are constituted in legal text. More importantly, Boyd White introduces the genre of regarding law as literature. Just as other literary works, legal texts behave similarly. However, he suggests that legal language is a specialized form, derived from its capacity for precision. Mellinkoff has, of course, attempted to debunk this myth. Nevertheless, Boyd White offers an alternative view. Namely, he is preoccupied with the *reputation* of precision around the language. In turn, he is focused on the conditions of the mind, and how language "demonstrates the condition of the imagination."[269] Rather than language as a tool of communication, the legal language is indicative of perceptual difference, a particular visualization of fact. Simply put, it is how the law sees the world.[270]

Consequently, Boyd White regards legal language less as a matter of expression, but, more so, as a relationship. Boyd White communicates this argument through his discussion of control. He argues

[269] James Boyd White, *The Legal Imagination* 6 (1973).

[270] There is overlap on the law's visualization of fact and the notion of law as "local knowledge" put forth by Clifford Geertz. He notes, "the vernacular characterizations of what happens connected to vernacular imaginings of what can. It is this complex of characterizations and imaginings, stories about events cast in imagery about principles, that I have been calling a legal sensibility..." *See* Clifford Geertz, *Local Knowledge: Further Essays in Interpretative Anthropology* 215 (1983).

that the legal language is inherited; a formal language that is imposed and has the quality of "defining the habitual expectations, the cast of mind, of the audience with which you will deal".[271] As opposed to choosing amongst questions of linguistic construction, the language has inherent limitations that require its adopters to master control in order for meaning to exist. He considers first metaphor as a form of control,[272] representative of depth and depiction of the inexpressible. He reflects on Joseph Conrad's *Mirror of the Sea*, an autobiography of Conrad's life written through the world of the sailor. In a similar light, he asks what may be the "world" of law and how is it written? Legal language can then be conceived as a metaphor. Language is not simply the law's functionary, but, in fact, a thumbprint of its social identity.

Equally, Boyd White points to ambiguity as a necessary counterpart to metaphor. That is, the use of metaphor requires accepting that there may be more than one meaning. Boyd White refers to *Moby Dick* as an informative example. The fixation on the whale and the inconsistency and variety of its depiction represents the pursuit for meaning, whereby "inherited systems of thought and language that give meaning to events no longer work."[273] Ambiguity enables the space for the uncontrolled, as no one meaning is settled. Moreover, Boyd White argues that giving meaning is not equivalent to explaining.[274] Rather, it is the opposite; language cannot explain, but can only afford particular significance to an experience. The legal language cannot articulate fact but can only read it.

The question becomes: how is the legal experience signified through its language? Marianne Constable returns to Austin and extends his theory to consider "legal speech acts."[275] Constable delves into the legal grammar, focusing on its "strange retrospective temporality."[276] She notes that law is "neither strictly causal nor chronological."[277] Written in the future perfect tense, the grammar indicates a commitment made in the present that refers to the future reflecting on a recent past. Retrospection and anticipation are inseparable. Equally, Constable points to the imperfect tense that

[271] Boyd White, *supra* 269 at 72.

[272] *Id* at 47.

[273] *Id.* at 58.

[274] *Id.*

[275] Marianne Constable, *Law as Language*, 1 CRITICAL ANALYSIS OF LAW 63 (2014).

[276] *Id.* at 68.

[277] *Id.*

is notable in legal utterances. She suggests that this is representative of the incompleteness of the law, knowledge that is interruptible and incapable of total attainment.

Constable's arguments are fascinating. She exposes the character of law as traceable in its grammar. This differs from prior discussions, as it appeared that the analysis was focused on the peculiarities of the vocabulary and sentence structure. These peculiarities were then assessed against interpretability and readability. Instead, the crux of her analysis centers on tense and construction of verbs in law's communicative function. Constable demonstrates that the legal grammar is an implicit representation of the law's behavior. Interestingly, her arguments are not persuasive in understanding how law is a language. Merely, she has reframed how the legal language is distinct in its form.

In contrast and reminiscent of Boyd White, Richard Posner refers to law as a literary medium. While both Constable and Posner reflect on the element of temporality, the intention is entirely opposite. Posner focuses on the temporal remoteness[278] as an explanation for issues of interpretation, whereas Constable sees it as a form of fingerprinting. Moreover, Constable does not reflect on meaning-making. Subsequently, Constable and Posner are a mirror to language and text as conversations of form and substance.

For Posner, reading legal text as literature reveals the law's intimate relationship with fiction. Throughout this chapter, legal fiction has been discussed on multiple occasions.[279] Posner, however, is not concerned with legal fiction as a feature of the legal language. In contrast, he considers that the law *is* fiction; and in effect, the legal language is figurative. Posner is not suggesting that there is no 'truth' to the text, but simply that tools, commonly found in literature, help generate fact in legal writing. Perhaps echoing Danet, the legal language is poetic and capable of painting the dissimilar as similar.[280] It is an imagination[281] built on metaphor, "an inescapable method by which we give structure to experience."[282] To substantiate his argument, Posner uses a seminal case on privacy, *Melvin v. Reid*.[283] Posner argues that as this case was initially tried on the merits, the factual recital, "as far as

[278] Richard A. Posner, *Law and Literature: A Misunderstood Relation* 15, 210 (1988).

[279] Recall in the discussion of Goodrich, Fuller, and Danet.

[280] *Id.* at 3.

[281] Posner frequently refers to Boyd White's text. See *id.* at 12.

[282] *Id.* at 4.

[283] 112. Cal. App. 285, 297 Pac. 91 (131). *See* also Posner's description of the "facts" of the case. *Id* at 4-5.

anyone knows," could have been fictitious.[284] Though the case was originally dismissed, the appellate court had requested it be tried again to determine whether the facts were as alleged. Interestingly, the tracks stopped there and there was no further trace of this case. Posner's example is representative of the body of judicial decisions that do not require verification of 'fact.' Rather, no one has ever known whether there was indeed an infamous snail in the ginger beer.[285] Still, these cases are "woven into the fabric of the law."[286] It follows that judicial decisions substantiate these narratives as truths.

Though the remainder of Posner's text is devoted to reconciling the lessons that may be learned between law and literature, he has offered a perspective on law as a linguistic conduit of reality. The law is a literary narrator and is, by design, built on fiction. It is like a ventriloquist; a performative experience that is false and consciously staged, nonetheless accepted on the basis of a circumstantial realism.[287] In consideration of law as language, how important is the use of natural language towards the success of the ventriloquist act? In other words, accepting the premise of law as communicating a reality, are these literary constructions (i.e., metaphor) wedded to its current form? The concluding section of this chapter strives to extend past the various conversations of law and language, confronting instead the intentions of expression and communication vis-à-vis thought.

An Ode to Natural Language: Constructing (Con)text

In analyzing the ways in which the relationship of law and language have been described, I identify two common aesthetics: (1) contour; and (2) shape. Contour represents the unique markers of the legal language; how each bend and curve distinguish it from another. Shape, on the other hand, represents legal language as a unique entity. It manifests itself and its surroundings bend to it. More importantly, contour and shape work in tandem. It may even be argued that these aesthetics are multiple sides of the same prism. Yet, the aforementioned scholars appear to be divided over underlying philosophical, linguistic, and literary reflections on the legal language. What is notable is the continued gap in literature on the role of *natural* language vis-à-vis legal language. Natural

[284] *Id.* at 5.

[285] *Donaghue v. Stevenson*, [1932] AC 562.

[286] Posner, *supra* 278 at 5.

[287] Francois Cooren, "In the Name of Law: Ventriloquism and Juridical Matters" in Kyle McGee (ed.), *Latour and the Passage of Law* 249 (2015).

language is perceivably accepted as a default tool for legal writing and a mere passing thought to their respective commentary.

Accordingly, the scholars fail to completely articulate the distinctiveness of natural language as the legal vessel. Together, however, they evidence that legal concepts have relied on the language for their expression and communication. That is, natural language has been the exclusive instrument for the law to conduct its work. What may be gathered are convincing arguments that justify the richness of the law's interpretative exercise. Directionality, however, has never been considered an issue. The closest critique may be found in Goodrich's discussion. To recall, Goodrich attempted to argue against the semiotics of legal argument. Though not his intention, Goodrich alludes to notions of conceptual transfer and intersubjectivity; language transports legal concepts that exist independently and merely find expression through its linguistic vessel. This suggests that regardless of the communicative tool, the legal concept could adapt accordingly. But, does natural language impact the construction of the concept? That is, would the legal concept exist if it was to be expressed in an alternative form?

Some scholars have argued that it could. Since the 1950s, Layman E. Allen had fervently argued for the use of symbolic logic in the expression of legal concepts.[288] Allen's specific arguments will be revisited in subsequent chapters. In short, he demonstrated symbolic logic was helpful to the extent of unpacking complex sentence structure. Nevertheless, there remained hesitations around the usefulness of symbolic logic for drafting.[289] These arguments largely center around the limits of symbolic logic in resolving legal complexity; that it was beyond a question of increasing precision, but simply that most ambiguity is unknown.[290] In other words, the law has an open texture and is inherently incomplete. Danet argued that the indeterminacy of the law is reflective of the indeterminacy of the language. Reframing her argument, could the indeterminacy of the language be, in part, the indeterminacy of thought, and specifically legal thought?

---

[288] *See* for example Allen's papers: Layman E. Allen, *Symbolic Logic: A Razor-Edged Tool for Drafting and Interpreting Legal Documents*, 66 YALE L. J. 833 (1957) and *Some Uses of Symbolic Logic in Law Practice*, 3 MODERN USES OF LOGIC IN LAW (1962).

[289] Consider the response from Robert S. Summers, *A Note on Symbolic Logic and Law*, 13 J. OF LEGAL ED. 486, 490-491 (1961).

[290] *Id.* at 492.

Accordingly, there must be a reflection on communication and the purpose behind its mechanics. Inevitably, Jacques Derrida comes to mind. Derrida considered the means of communication, and specifically the mode of writing.[291] He questions whether there is a "homogenous space of communication" that writing is capable of extending.[292] He retraces the origins of writing, noting that "thought" was regarded as preceded and governed communication.[293] Writing, then, is perceivably a means of transmitting thought; the transmitter is independent of what is being transmitted. Derrida suggests that the structural characterization of writing as representation– and thereby its mechanical character – offers the impression that the relation between idea and sign (words) could never be "either annulled or transformed."[294]

The problem, Derrida argues, is the notion of absence. Unlike other forms of communication, the "speaker" is absent. Writing is "the mark that he abandons, and which cuts itself off from him and continues to produce effects independently of his presence and of the present actuality of his intentions."[295] Written communication, then, has the quality of permanence. Its structure inherently enables outliving its author and the original linguistic and cultural context. Derrida describes this as the "breaking force" that ruptures context.[296] Interestingly, this 'removal' of context does not preclude the readability of the sign. Instead, it marks the ability for writing to be grafted. Derrida states, "no context can entirely enclose it. Nor any code [...]"[297]

The possibility of disengagement and grafting is further demonstrated in citation. Derrida highlights that if placed between quotation marks, there enables an infinity of new contexts in a manner which is absolutely illimitable."[298] So what does this mean? The capacity for the unlimited carving of text, and subsequent mutability to other text, describes the directionality of language impacting thought. In turn, concepts cannot be extracted from natural language as they are not encased by it. Derrida suggests that written communication then is not a vehicle for the "transference of meaning;" meaning

[291] Jacques Derrida, "Signature Event Context" in *Limited Inc.* 2 (1977).

[292] *Id.* at 3.

[293] *Id.* at 4.

[294] *Id.* at 5.

[295] *Id.*

[296] *Id.* at 9.

[297] *Id.*

[298] *Id.* at 12.

is a mere effect of writing.[299] Perhaps the legal historians Mellinkoff references were right: law has indeed been subjected to its writing. Derrida concludes, "deconstruction does not consist in moving from one concept to another, but in reversing and displacing a conceptual order as well as the nonconceptual order with which it is articulated."[300] Paradoxically, the simultaneous inability to anchor context and ability to graft text destroys the separation between the casing and encased.

I, therefore, draw two possible conclusions: (1) natural language is the only vessel in which legal concepts may be housed; or (2) an alternative vehicle may be able to house legal concepts, on the premise that it must inherit natural language's traits. The former may be framed in the guise of Agamben's arguments. That is, natural language is the law's signature. The perspective of the latter is less absolutist, and more nuanced. It suggests that, even in accepting the deconstructionist view, there must necessarily be a mirroring, and at minimum, mapping of the ways in which the concepts have taken shape through writing. More importantly, it is a test against the limits of written legal expression. Regardless, both conclusions arrive at an inherent need to unpack the linguistic construction of natural language to better understand the law's embedded code. The following chapter applies the considerations of this chapter and explores in depth the various pillars of linguistics.

[299] *Id.* at 20.

[300] *Id.* at 21.

2- Language Lego

Irrefutably, there is a bond between law and language. In the prior chapter, there had been discussion at length on the role and significance of language in legal text. Regardless of how the relationship between law and language is perceived, the traditional understanding of their relationship has not considered in depth the analytical weight of linguistics. However, several legal scholars have provided grounds for further linguistic investigation. Peter Tiersma alluded to the uniqueness of the legal language, an entirely separate language with its own linguistic constraints. Along a similar path, Marianne Constable reflected on the specific grammar choices in legal language. Notably, Constable described the choice of verb tense as characteristic of law's open texture. Brenda Danet, on the other hand, introduced a more nuanced practice. That is, in order to recognize how language interacts with law, there must necessarily be a venture into the linguistic makeup. I describe here core linguistic practice, often referred to as the "science of language."

Putting forth the argument that methods of core linguistics must be examined to better understand its legal impact, this chapter intends to walk through three essential pillars of natural language: (1) syntax; (2) semantics; and (3) pragmatics.[301] The mechanics of how natural language is shaped and deconstructed provide an insightful commentary on existing understandings of meaning.

Furthermore, this section is an exploration of the known "subfield" of linguistics: computational linguistics. These methods are frequently used to translate natural language to computer code. More specifically, computational linguistics is understood as mirrors to linguistic methods of treating natural language. But, as opposed to allowing natural language to be understood by humans, these techniques allow natural language to be understood by machines. It is then another intention of the chapter to investigate whether they are, in fact, functional equivalents.

This section will unfold as follows. Starting with syntax, the chapter will introduce core tenets of sentence structure, diving into generative grammars, constituents, and dependency trees. The chapter will then advance into meaning, specifically how meaning is formed. Semantics views the meaning of sentences as sets of worlds that share the same truth conditions. Pragmatics, on the other hand, factors the context inferred and the accounts of "additional meaning." While the former is built on propositional calculus and predicate logic, the latter is built on implicature, reference, and

[301] It is important to note that these are not the only subfields of linguistics. There are several others, but for the intentions of the dissertation, they will not be discussed.

presupposition. In short, semantics is predominantly context-independent while pragmatics is context-dependent.

Alternatively, their counterparts in programming will be considered; beginning with regular expressions and context-free grammars designed for syntax. The section will subsequently progress into attribute grammars. These tools are often used to provide context-sensitivity when defining the semantics of a programming language. Perhaps the most exciting discussion will turn to abstraction and logic programming used to classify and conceptualize worlds. From the fundamentals, the chapter will turn to knowledge representation and complexity.

The aim of this chapter is to garner a deeper understanding of linguistic tools and to engage with notions of computation through an unconventional framing. I hope to redirect the focus from computational linguistics to computation *and* language. More importantly, the section will act as a primer, helping to bridge disciplines and engage in more complex investigations around the translation of law to code. I must provide the disclaimer that I am neither a linguist nor a computer scientist. I lean on texts that have been described as foundational to these disciplines. As well, the chapter certainly does not and cannot claim to be exhaustive. Its intention is merely to introduce and provide the foundation and lens for analysis. Consequently, I thank immensely fellow colleagues who have helped educate, inform, and verify the material I present.

Syntax: Sentence Architecture and Structural Integrity

Syntax studies form, and more specifically, the organization of words to sentences. Syntax is frequently conceived as embodying a cognitive component, as its theories consider how words are generated from abstract thought to sentences. It follows that the leading syntactic theory[302] is known as Generative Grammar, developed by Noam Chomsky in the 1950s. The underlying thesis is that sentences are produced through a subconscious set of procedures[303] and that syntax is simply a model of this process. Syntax is preoccupied with the formal properties of language and observes them through a scientific method. It involves gathering mass empirical data and building generalizations,

[302] I, importantly, acknowledge that Chomsky has received over the years criticism in his work on Generative Grammar, specifically his notion of innate models. *See* for example, Paul Ibbotson and Michael Tomasello, "Evidence Rebuts Chomsky's Theory of Language Learning," *Scientific American* (Sept. 7, 2016) available at: https://www.scientificamerican.com/article/evidence-rebuts-chomsky-s-theory-of-language-learning/. I maintain that, for the purposes of providing a foundational, introductory perspective on syntax, it is nevertheless an informative starting point.

[303] Andrew Carnie, *Syntax: A Generative Introduction* (3rd ed. 2013).

then drawing hypotheses accordingly. A syntactic hypothesis is defined as a rule and a group of hypotheses is understood as a grammar.[304] Syntactic models then carry a set of grammatical rules that inform of acceptable word order. As a result, this ordering generates sentences. Again, these steps are procedural. As syntax is perceivably a model of producing language, these rules are also descriptive.

Grammaticality investigates the acceptability of a sentence on the basis of a competence-performance distinction.[305] Competence considers whether a sentence is interpretable in a language; effectively, whether the sentence is well-formed. In contrast, performance refers to the act of executing a language, the real-world behaviors that are a result of language knowledge. Therefore, acceptability from a syntactic perspective focuses on competence. Acceptability is entirely structural and associated with the "mental ability to break apart sentences."[306] Parsing sentences – deconstructing phrases into bits – has certain limits, and these limits affect whether sentences may be interpretable.

For Chomsky, the parsing exercise is innate to human language generation. Chomsky raised a distinction between Language (with a capital L) and language (with a lower-case l). Language (with a capital L) is the cognitive capacity to create language (with a lower-case l). On the other hand, language is an instantiation of this ability.[307] Language is instinctual and built into the human brain. This facility is known as Universal Grammar (UG).[308] UG is described as a "flexible blueprint" for constructing the knowledge of language.[309] It constrains the processes that "map between situations and utterances."[310] UG also enables recursion; an ability to embed structures iteratively and produce infinite possibilities of sentences, even if they have never been generated before.[311] Equally, human language shares certain properties, the same basic innate materials for building a language's grammar. This 'built-in' system has core, atomic components for generating sentences. The acquisition of

[304] *Id.* at 8.

[305] *Id.* at 17.

[306] *Id.*

[307] *Id.* at 5.

[308] *Id.* at 19.

[309] *Id.* at 23.

[310] *Id.*

[311] *Id.* at 33.

language can then be reduced to the "setting of certain innate parameters."[312] For instance, the setting of the subject-verb-object (SVO) order. Though there are variations in how they are to be ordered, this is one common arrangement. Fundamentally, the treatment of the parameters and how they are set belong to the broader approach to syntax. Furthermore, it relies on the assumption that certain grammars are inherent to the human brain and the rest is acquired.

In short, syntax is the study of sentence structure. While syntax considers, in part, the intrinsic competence to generate acceptable sentences, syntax also reflects on sentential architecture. Words in a sentence may be grouped into units called "constituents" that function together.[313] These constituents then are embedded into one another to form larger constituents, described as "hierarchical structure."[314] These larger constituents eventually form sentences. It is perceivably an assembly line for words and parts of words. Syntax considers the "purely intuitive level" of how words appear to be related to one another. These intuitions are captured by the notions of constituency and hierarchical structure. Sentences in generative syntax are represented in the form of a hierarchical tree structure, illustrating the relationships between constituents.

In generative grammar, structure is represented by rules. The basic set of rules is known as phrase structure rules (PSRs). These rules are one method of breaking down sentences to consider their component parts. They reveal the manner in which phrases embed themselves and the structures that allow for grammaticality. Below is a generalized list of PSRs:[315]

a) CP → (C) TP
b) TP → {NP/CP} (T) VP
c) VP → (AdvP+) V (NP)({NP/CP}) (AdvP+) (PP+) (AdvP+)
d) NP → (D) (AdjP+) N (PP+) (CP)
e) PP → P (NP)
f) AdjP → (AdvP) Adj
g) AdvP → (AdvP) Adv
h) XP → XP conj XP
i) X → X conj X

[312] *Id.* at 28.

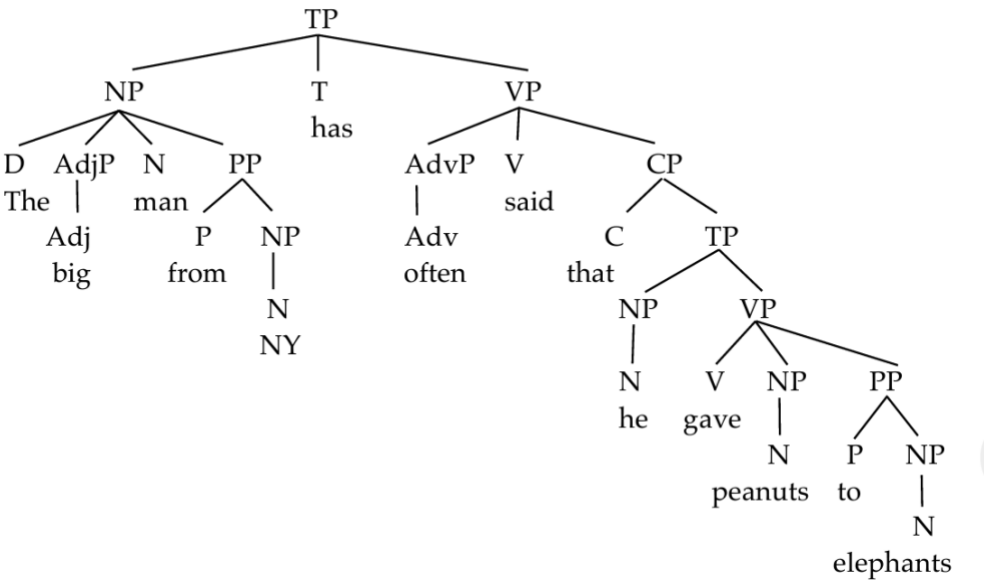
[313] *Id.* at 72.

[314] *Id.* at 73.

[315] *Id.* at 89.

An initial observation is the mathematical nature of PSRs. The variables in PSRs all represent various parts of speech (e.g., nouns, verbs, prepositions, etc.), with arrows representing how and when these variables combine to become phrases. As an early Chomskyan approach, PSRs operate such that application of these rules account for the formation of any English sentence. While PSRs are a typical starting point in understanding syntax, PSRs, as a method, were soon overtaken by constituency grammars like X-bar theory and Minimalism. Nevertheless, their fundamental ideas remain inherently unchanged. Below is an example of how a sentence would be rendered into a tree structure:[316]

64) The big man from NY has often said that he gave peanuts to elephants.



Syntactic trees also play an important role in unpacking ambiguous sentences. Consider the phrase:

Elaine ate the pasta in the kitchen.

This sentence is structurally ambiguous as it could mean either (a) Elaine ate the pasta that was sitting in the kitchen; or (b) Elaine ate the pasta and did so in the kitchen. Both meanings of these sentences are equally possible, owed to the principle of modification.[317] The first meaning has the prepositional phrase (PP) *in the kitchen* modifying the noun *pasta*. The PP describes which pasta. It modifies the

[316] Example taken directly from Carnie. See *id.* at 90.

[317] *Id.* at 96.

noun and is considered part of the noun phrase (NP). In the latter case, the PP *in the kitchen* modifies the verb *ate*. The PP describes where the pasta was eaten. It modifies the verb and is considered part of the verb phrase (VP). In short, the notion of modification is one example of how structural relations between words alter its meaning. However, it does not indicate how to determine meaning, but simply that there is more than one.

As mentioned, the rules that guide phrase structure composition depicts the mathematical properties of syntax. The internal structural relations are generalizable and support how sentences are pieced together. Just as syntax informs how sentences are assembled, it equally informs of the constraints of assembly. For instance, a locality constraint is the rule that two syntactic entities must be near one another. Two important notions must be discussed: (1) coindexation; and (2) binding. Coindexation refers to the structural relationship between nouns in a sentence. An NP that gives meaning to another NP is described as the relationship between antecedent and anaphor.[318] For example, consider the sentence:

> The woman (*antecedent*) was proud of herself (*anaphor*).

A personal pronoun, on the other hand, is an NP that may derive meaning from another word in the sentence, or from context and previous sentences in a given text.[319] Coindexation refers to the notion of marking when two NPs refer to the same entity. See for example:

> [Adam]$_i$ claimed [he]$_i$ went to the library yesterday.

Two NPs that are coindexed are also described to corefer. Coindexation, or coreference, reveals that, within syntactic hierarchical structures, anaphors or pronouns must accord with certain conditions vis-à-vis the antecedent. This is known as binding. Consider the below sentences:

> Hannah wrote herself a letter.
> Hannah's mother wrote herself a letter.

Both sentences have NPs that are coindexed. The difference, however, is the coindexing of the anaphor *herself*. It is clear that *herself* refers to *Hannah* in the first sentence and *Hannah's mother*

[318] *Id.* at 150.

[319] *Id.* at 149.

in the second. But, how do speakers know the distinction? Why is it ungrammatical for *herself* to mean Hannah in the second? Consider alternatively the sentence:

Hannah's mother admires her.

In this case, the pronoun *her* is coindexed with *Hannah*. Binding theory sets out to specify the acceptable options relating to antecedents and their coreferents. A simple set of binding principles govern coindexing. In accordance with *Binding Principle A*,[320] anaphors must be bound in their binding domain. On the other hand, *Binding Principle B* applies to personal pronouns; that personal pronouns must not be bound in their binding domain.[321] Any other type of noun generally is "unbound" by nature. Binding domain is generally understood as the boundary between constituents that contain the antecedent, loosely interpretable as the clause in question.

The notion of coindexation, though intuitive to native speakers, is, in fact, incredibly complex to describe syntactically. Nevertheless, these concepts become important in the consideration of how legal texts are written. In particular, legal concepts are often referenced in a manner that muddies the structural hierarchies and relationships within sentences.[322] Ultimately, the discussion on syntax, and in effect, generative grammars, centers on structure and form, embodying innate mechanisms. In contrast, there is little to no discussion on content. Meaning is broadly presumed as separate from syntax, with the exception of clarifying constituent relationships. The next section will advance past structural to substantive investigations.

Semantics: To Mean or Not to Mean

In the prior chapter, meaning was a recurring motif across the analysis of law and language. This is, of course, no surprise as legal analysis centers on the interpretation of words. As discussed, meaning is rather elusive. There is often a devotion to definition, wholehearted attempts to secure parameters and pin down words. Dictionaries are considered as sources of references but could only provide hints and not conclusive meaning. For linguists, "defining the meaning of a word is an enterprise of

[320] *Id.* at 157.

[321] *Id.*

[322] *See* for example the statement, "The phrase 'carries a firearm' applies to a person who knowingly possesses and conveys firearms in a vehicle, including in the locked glove compartment or trunk of a car, which the person accompanies." It is not clear what the relative pronoun *which* is referencing. See *Muscarello v. United States*, 524 U.S. 125 (1998).

almost inconceivable complexity."[323] More importantly, definitions are only a microcosm of meaning. The process of uncovering meaning is far more arduous. So, what does it mean to mean?

There are broadly two categories of meaning: (1) intention-free indication, or natural meaning; and (2) indication-free intention, or non-natural meaning. The former is a state of existence. The relationship "just is."[324] The latter is more interesting. Non-natural meaning builds a connection that is intentional; it was decided that one thing will stand for another. It is neither automatic nor intuitive. This is frequently described as the relationship between form and content and where language exists. Interestingly, within non-natural meaning, there are two variants: (1) non-linguistic; and (2) linguistic meaning. While this section will largely discuss linguistic meaning, it becomes clear that non-linguistic meaning plays a heavy role in the advent of computation.

Linguistic meaning describes the arbitrary relationship between most words and what they represent.[325] As well, meaning is composable. That is, there are various units, each embodying their own meaning, that may be pieced together to create another meaning. Described here is the concept of stringing words to construct sentences. There is seemingly overlap between syntax and semantics; and to a certain extent, syntax already articulates how form and substance are perceivably distinct. Consider the sentence famously used by Chomsky:

> Colorless green ideas sleep furiously.

The sentence bears no content, but its structure is entirely correct. This sentence continues to stand as a fantastic example of how syntax and semantics play different roles in natural language understanding. Namely, a clear distinction between semantics and syntax is the preoccupation with compositional creation of meaning, as opposed to the interaction between structural arrangement and substance. That is, semantics reflects on how the form of the sentence informs how meaning of words may be "built up into the meanings of sentences."[326] In accordance with sets of rules, larger meanings are made possible by smaller meanings. It is an investigation on how the literal meaning

[323] Paul Elbourne, *Meaning: A slim guide to semantics* 1 (2011).

[324] Betty Birner, *Language and Meaning* 3 (2018).

[325] *Id.* at 4.

[326] *Id.* at 9.

of a sentence depends on the semantic meaning of its component words and how those words may be woven together.[327]

Referencing Chomsky's sentence, a syntactic perspective would note that the structure is unambiguous and, therefore, the sentence is clear. From a semantic perspective, understanding the evident paradox between *colorless* and *green* would immediately signal that this sentence is non-sensical. Coupled with the understanding that ideas cannot sleep, nor in a manner that is furious, this sentence becomes utterly meaningless. Through this example, it follows that semantics is focused on the study of conditions and the relations with which meaning may be established.

A dominant theory[328] within semantics is truth-conditional semantics. The notion is that the meaning of a sentence is the set(s) of worlds in which it is true. Otherwise, the meaning of a sentence is "the proposition it expresses;"[329] whereby propositions are considered sentences that can either be true or false. Truth-conditional semantics articulates the procedure for determining meaning and categorizing when it does or does not apply. Referring again to Chomsky's sentence, the word *idea* represents a particular set of individual objects and carries certain traits. These traits distinguish *ideas* from other objects, such as *chairs*. Consequently, what is an idea is, in fact, what are the conditions for a given object to be an idea.

This logic extends from words to sentences. The conditions under which a word or sentence are true are known as truth-conditions. The truth-value of a word or sentence is simply whether the sentence is true or false. These two terms are important as truth-conditions are absolute in all worlds, whereas truth-values are relative to the world. Importantly, semantics borrows from the study of logic, effectively representing meaning in terms of truth.[330] The meaning of words can then be regarded as how they affect the truth-conditions of a sentence.

Consider a simple example: what constitutes as a sandwich? Though this appears straightforward, what may be its truth-conditions? Interestingly, this discussion was brought before the Massachusetts Superior Court ("Court"), seeking to determine whether a burrito was a sandwich. In 2006, White City Shopping Center ("White City") sought a declaratory judgment that it was not in violation against

[327] *Id.* at 12.

[328] Here I am using the word "theory" as akin to method(s) as opposed philosophy.

[329] Birner, *supra* 324 at 39.

[330] *Id.* at 40.

the commercial lease signed with PR Restaurants ("Panera"), a company that operates Panera Bread restaurants.[331] For context, Chair 5 restaurants, operator of Qdoba restaurants, wanted to open an outlet in White City. Qdoba is a Mexican restaurant chain that sells burritos. However, the commercial lease between White City and Panera contained an exclusivity clause preventing White City from engaging in agreements with restaurants that would directly compete with Panera's sandwich sales.

So, is a burrito a sandwich? Panera had argued that any food product with bread and a filling is a sandwich. According to the Court, it is not. From a semantic perspective, what set of objects does the word *sandwich* denote? Again, the traits are important in the classification of an object. Componential semantics regards the "set of primitive features that an object either must have or must not have in order to count as an instance of that term."[332] A simple example would be the word *child*.[333] The deconstruction would look as follows:

+human
– adult

This denotation is to represent that a child is a human that is not an adult. Using this methodology, a sandwich may be broken down into the following:

+bread

This is problematic as a further assessment requires understanding the primitive features of bread. Returning to the construction of a burrito, would tortilla be considered bread?[334] Is the primitive feature of bread +flour? These questions reflect the lack of clarity involved in componential semantics and the vicious circle involved in breaking down seemingly basic words. To resolve this conundrum, linguists often turn to prototypes, the archetypal example of a particular word, as a

[331] *White City v. PR Restaurants*, No. 2006196313 (Mass. Cmmw. Oct. 31, 2006).

[332] Birner, *supra* 324 at 52.

[333] Example directly taken from Birner, *id.*

[334] Panera put forward the argument that tortilla qualifies as bread. However, the Court ruled that this argument was misplaced, as the ordinary meaning applies when interpreting unambiguous contractual terms. The Court argued that Panera did not provide evidence that the term "sandwiches" intended to include burritos. See *White City v. PR Restaurants*, *supra* 331.

reference point. The more similar the object is to the prototype, the "more properly" the word applies to it.[335]

Prototype theory relies on a core and periphery analysis in the assessment of meaning. The prototype lies at the center and decreasing similarity borders into the territory of it not being the object. Evidently, the application of prototype theory suggests that truth values may not be a clear true/false binary. More importantly, this also suggests that truth conditions may be blurry. Linguists often discuss the parallel between prototype theory and the notion of fuzzy logic.[336] The idea is that meaning is captured on a spectrum and a matter of degree. The understanding of the word is dependent on a process of continual refinement and a statistical calculation of likelihood. This discussion will resurface in a later chapter.

Beyond complexities in establishing individual word meaning, semantics equally considers the relationships between words and sentences. First, there are a number of a ways that words can relate to one another, and each correspond to a particular aspect of meaning. A few key relationships will be discussed here: synonymy, homonymy, polysemy, and metonymy. I have elected to select these concepts as they most reflect the linguistic issues in legal texts. To start, synonymy is the relationship between two different forms with the same meaning. Again, the form and content divide resurfaces. Synonyms also reflect similar, but not identical functions. Slight differences persist and reinforce the aforementioned issues discussed on classification. On the other hand, homonyms are two identical forms with different meaning. Homonyms introduce ambiguity, defined by linguists as having "more than one distinct meaning."[337] It is important here that ambiguity is discussed separately from vagueness. A word is vague if it has a meaning "that does not distinguish between two or more different kinds of things."[338] While they often appear in tandem, they are not, in fact, the same semantic property.

As a result, an associated concept is polysemy. Polysemy may be considered as homonyms on a gradient scale. That is, polysemous pairs are also two identical forms with different meaning, but that these meanings are related. Consider the word *glass*. Between a glass of water and the material glass,

[335] Birner, *supra* 324 at 53.

[336] *Id.* at 54.

[337] *Id.* at 59.

[338] Elbourne, *supra* 323 at 34.

they both share a common makeup but reflect two different meanings. Whereas homonyms have entirely distinct meanings, polysemous pairs have relatively different meanings. Finally, metonymy is perhaps the most complex. It borders on metaphor but is a word that represents a closely related concept. For example, the Crown is often used to represent Queen Elizabeth II or, more broadly, sovereign power, in comparison to a crown describing an ornamental headdress.

These relationships suggest that no two words are truly alike, neither in form nor substance. Should there be exact duplicates in meaning or function, linguists suggest that one "would die out, since the need to learn and remember two words for the same thing puts an unnecessary burden on the language user."[339] This is fascinating, as it implies that inherent to natural language is an evolutionary Darwinism such that, in spite of similarity, there cannot be singularity for the very reason that exact variations would simply not survive.

Just as relationships between words help ascertain meaning, relationships between sentences are likewise significant. Hyponymy is the notion of subcategories and belonging to the same 'family' of concepts. Consider the words *rose* and *flower*. A rose is a flower but is a specific type of flower. Hyponymy then demonstrates a taxonomy[340] between words and a hierarchy of understanding. At the sentential level, hyponymy parallels entailment; for one sentence to be true, the other must necessarily be true.

Consider the following:

> Megan is shorter than William, and William is shorter than Ryan.
> Megan is shorter than Ryan.

These two sentences entail one another, as the truth-conditions of the former necessitate the truth-conditions of the latter. In other words, Megan must be shorter than Ryan as she is already shorter than William. Though not explicit, the meaning of the first sentence is inclusive of the second. More importantly, entailment may be regarded as the central notion in truth-conditional semantics.

As discussed, sentence meanings are drawn from word meanings.[341] This is particularly noticeable with ambiguity. That is, lexical ambiguity gives rise to sentential ambiguity. Should a word within a

[339] *Id.* at 56.

[340] *Id.* at 58.

[341] Birner, *supra* 324 at 62.

sentence be ambiguous, the entire sentence is potentially ambiguous. This is understood as the notion of compositionality. First introduced through homonyms, semantic ambiguity describes the possibility of a single form with multiple meanings. Semantic ambiguity, however, also includes structural ambiguity at the sentential level. Linguists often refer to this example to represent both types of semantic ambiguity occurring simultaneously:

> Time flies like an arrow; fruit flies like a banana.

This sentence is a play on both lexical and structural ambiguity. First, the words *flies* and *like* are lexically ambiguous. *Flies* is used as a verb in the former and noun in the latter. *Like* bears the meaning of "similar to" in the former and "fond of" in the latter. Structurally, the former phrase splits between *time* and *flies*, whereas in the latter, the clause splits at *fruit flies* and *like*. The difference in structure renders the second phrase initially ambiguous. Notably, both syntactic and semantic ambiguity contain structural ambiguity. This again plays a role further in the chapter.

As will be seen, programming languages draws inspiration from numerous concepts in both semantics and syntax. In order to better understand these parallels, it is important to build a foundation on the semantics metalanguage – how linguists symbolically represent semantic meaning. The claim is that the metalanguage not only allows linguists to circumvent "the ambiguities of natural language," but also enable the "representation of each meaning of an ambiguous sentence."[342] This is perceivably a "one-to-one representation" between the notation and meaning. Working through the basics of both syntax and semantics, logic is an evident undercurrent of the discipline. While the chapter will not delve into the specifics of the semantic metalanguage, there will be discussion on its most important concepts.

For linguists, verbs are the hearts of sentences.[343] As a result, sentences typically pivot around the verb. Semanticists use the term predicate[344] to describe the verbs. Predicates are then considered functions that operate on sets of objects. These sets are known as a domain. The function informs of the objects within the domain. Its performance determines the truth value of the resulting proposition. These terms and understandings are evidently drawn from formal logic.

---

[342] *Id.* at 75.

[343] *Id.* at 69.

[344] Note that this term is not to be equated with those in grammar or syntax.

Consider the following metalanguage translation of an ambiguous restaurant menu option:[345]

> Natural language: Customers may have soup and side salad or salad bar.
> Metalanguage:
>
> $$(\text{soup} \wedge \text{side-salad}) \vee \text{salad bar}$$
> $$\text{soup} \wedge (\text{side-salad} \vee \text{salad bar})$$

While this is a simplistic representation, it describes how sentences may be broken down into the potential variants of their meaning. The $\wedge$ and $\vee$ are evidently symbolic shortcuts for "and" and "or" respectively. Other examples of metalanguage notation include $\forall$ for "all" and $\exists$ for "there is at least one" or "there exists." Together, they act as a universal set of symbolic representations to interpret natural language sentences.

A sample sentence may be denoted as:[346]

$\forall x \exists y(L(x,y))$ 'Each person loves another person'

The sentence expresses that 'For all of *x*, there's a *y* such that *x* loves *y*.' This representation is a translation of the natural language version, 'Each person loves another person;' otherwise, one possible meaning of 'Everyone loves someone.'

The metalanguage is observably composed of logical operators. Its intent is to both identify and represent variable strains of meaning. This suggests that not only is semantics derived from mathematics, but that it remains a core basis of its analysis. More importantly, semantics relies on propositional truths. Once a particular world is established, meaning rests within the specific realm of truth in this world. Semantic meaning is then a mathematical manipulation of truth conditions, which do not extend beyond the relations of its words and sentences. The problem is that there may be more to meaning than what a simple true/false binary could convey. This may be particularly important in the consideration of legal texts, as the law frequently traverses past factual to account for normative constructions.

[345] Example taken directly from Birner, *supra* 324 at 76.

[346] Example taken directly from Birner, *id.* at 77.

Consider the following sentence:

> Elizabeth thinks that the tax policy is unjust.

From a purely semantic perspective, meaning predicates on the truth of Elizabeth's claim and not on the content of it. This means that the truth-value of the sentence solely depends on the facticity of the belief. That is, does Elizabeth actually think the tax policy is unjust? Whether or not the tax policy is, in fact, unjust is irrelevant. Interestingly, equating meaning as sentence truth suggests a subversion of an embedded truth. This is referred to, in linguistics, as "opaque contexts."[347] How then does one transcend past sentential meaning to meaning that is inclusive of and sufficiently captures context? Pragmatics, therefore, becomes fundamental in linguistic analysis as it brings to light questions of interpretation and intention.

Pragmatics: Is that what it means?

Semantic meaning struggles to establish the logical meaning of connectives. Conjunctions, such as *and*, have the potential of revealing meaning beyond lexical and sentential truths. As discussed, natural language often contains meanings that are subtextual, or express more than what is stated. H.P. Grice's seminal paper, "Logic and Conversation," articulates a theory to bridge between semantic and additional meaning.[348] In fact, his paper became the foundation for pragmatics: the study of language in context.

Grice argues that natural language embodies both elements of convention and intention. Convention is, broadly, the semantic focus; a deduction of what the word or sentence typically means. Notably, convention suggests context independence and that the logic to formulate meaning is rather universal. On the other hand, pragmatics is context specific, prioritizing intention and the role of the speaker. To reconcile convention with intention, Grice puts forward the "Cooperative Principle."[349] The Cooperative Principle stipulates four categories of maxims that describe the relationship

[347] *Id.* at 91.

[348] H.P. Grice, "Logic and Conversation" in Cole et al (eds.), *Syntax and semantics 3: Speech Arts* 41-58 (1975).

[349] *Id.* at 45.

between convention and intention: (1) Quantity; (2) Quality; (3) Relation; and (4) Manner.[350] The maxims are as follows:[351]

Quantity

1 Make your contribution as informative as is required (for the current purposes of the exchange).
2 Do not make your contribution more informative than is required.

Quality: try to make your contribution one that is true

1 Do not say what you believe to be false.
2 Do not say that for which you lack adequate evidence.

Relation

1 Be relevant.

Manner: be perspicuous

1 Avoid obscurity of expression.
2 Avoid ambiguity.
3 Be brief (avoid unnecessary prolixity).
4 Be orderly.

Effectively, these maxims imply that cooperation is the key ingredient that bridges between what is stated to what is meant. Language operates as a mutual and recursive form of understanding, premised on latent shared conventions and expectations around interpretation.[352] Equally, this suggests a duality in the formation of meaning; that expression necessitates communication. More importantly, fulfilment of these maxims illustrates how meaning traverses past sentential to additional. This is predominantly done through implicature. For linguists, implicature is similar to entailments. They are logically valid conclusions that are not stated outright but can be inferred from what has been stated.[353] Returning to the notion of logical connectives, consider the below sentences:[354]

> Brenda had charcuterie and cheese.
> Brenda had charcuterie or cheese.

[350] *Id.* at 45-46.

[351] Grice's Cooperative Principle maxims captured from Birner's summary. *See* Birner *supra* 324 at 97.

[352] *Id.* at 134.

[353] *Id.* at 99.

[354] Drawn from Birner's example. See *id.*

Hypothetically, this may have been a situation whereby a dinner host asks what Brenda had eaten. From a logical and semantic perspective, the word *or* appears to be inclusive. That is, both statements are necessarily true because it is known that Brenda had at least one of these foods. However, the second sentence used in natural language, in fact, implies exclusivity. That is, it suggests that while Brenda did consume these items, it is unknown which of the two. A response of the former, in compliance with the maxims of Quantity and Quality, would indicate a sense of certainty that Brenda had consumed both items. Consequently, the use of *or* would otherwise be unnecessary unless the speaker was not certain. As a result, Grice demonstrates that connectives can exhibit both their logical meaning and their potentially polar use in natural language; in effect, how intention may be conveyed in text.

Interestingly, flouting these maxims also reveals a divergence from logical meaning. Consider the use of a literary device, such as irony or metaphor. In accordance with the maxim of Quality, a violation would occur through statements that are blatant falsities and that stray from literal truths. Nevertheless, the intended meaning remains true. Therefore, despite literal meaning being false, context guides its interpretation and enables its communication. As discussed in the prior chapter, legal texts frequently depart from literal meaning. The language is often laced with metaphor and other literary devices. This will become important, particularly in the context of how legal text is translated from natural to programming language.

In short, implicature highlights what is not explicitly uttered. Moreover, it further demonstrates that utterances serve a purpose that extends beyond their logical expression. There is presumably a motivation behind their formulation. As well, implicature articulates one of the reasons for multiple interpretations of meaning (note the distinction with multiple meanings). Because meaning is inferred from performance, there is an inevitable gap between intention and interpretation.

In addition to implicature, pragmatics also considers the notion of reference. That is, descriptions often fail to accurately make reference to objects. Consider the following example:

> She is a renowned Supreme Court Justice.

The referent *she* is a noun that, semantically, could be used in many cases. However, only through context could an "obvious target for the reference"[355] be revealed. Occasionally, clarifications of the referent *she* introduces further ambiguity.

> Sonia Sotomayor is speaking with Natalie Leung.
> She is a renowned Supreme Court Justice.

Unless one has the active knowledge of who Sonia Sotomayor is, the referent *she* remains semantically unclear. Suppose that one does not possess this particular piece of information, *she* could then be referring to either Sonia or Natalie. This subsequently leads to an issue with regards to meaning making. It is often presumed that the truth-conditional semantic meaning of a sentence is determined prior to applying context clues (i.e., Grice's Cooperative Principle).[356] That is, semantic precedes pragmatic analysis. In the above example, the order of this process does not work. This is because the sentences are only true in one case and not the other. As a result, there is a necessary determination of the referent *she* – the context and intended meaning – in advance of establishing the truth-value of the sentences. This again reasserts that pragmatics is not a separate pillar of meaning, but conversely, interwoven to it.

Perhaps the most fascinating discussion on reference centers around the definite article *the*. Linguists often describe the use of *the* as "remarkably complicated," as it reveals the difference between implicit and explicit knowledge.[357] Unlike most words, the use of the definite article is entirely dependent on context. Often, *the* is a marker for precision, as is typically found in legal documents. Linguists find that the most common theories on definiteness appeal to the properties of familiarity and uniqueness.[358] Should a referent be both familiar and uniquely identifiable, the definite article will likely be used. Oddly, though familiarity and uniqueness are reasons for the use of *the*, it is neither necessary nor sufficient in explaining why the definite was chosen over the indefinite article. Simply, the definite is more appropriate than the indefinite. Consider this example:[359]

> The fastest way to get downtown is to take the train.

---

[355] *Id.* at 108.

[356] *Id.*

[357] *Id.* at 109.

[358] *Id.* at 110.

[359] Example taken directly from Birner. See *id.* at 112.

There is no intention to specify any particular train. Instead, it alludes to a "complete irrelevance of the identifiability of the particular referent."[360] It matters more the category, as opposed to the particular member of the category.[361] Therefore, the definite article is a linguistic enigma that poses challenges not only on rules of its usage, but more broadly, its purpose.

This complexity with defining rules around definiteness bleeds into another concept within pragmatics: presupposition. Presupposition is interesting, as it muddies the boundary between semantics and pragmatics. Presupposition is understood as implicit information that is often taken for granted.[362] Consider the following sentence:

> Jonny's brother is a legal engineer.

A presupposition is as simple as the implicit assumption that Jonny has a brother. Two related concepts emerge: (1) semantic presupposition; and (2) conventional implicature. A sentence presupposes a proposition if the proposition must be true in order for the sentence to have a truth-value. Semantic presuppositions follow a "three-valued logic."[363] As opposed to only having two values (true or false), there is third possibility of being neither. That is, a proposition enables the sentence to be either true or false. Consider the below example:

> If Alex has a car, he will not mind working far away from home.
> Alex works far away from home.

The proposition that Alex works far away from home must be true in order for the sentence to have a truth-value. Conventional implicature, on the other hand, is considered as "species of entailments," which arise from the particular choice of words or syntax.[364] Often equated with semantic presupposition, the information conveyed in the expression sufficiently provides the context inferred. Consider the example:

> She has not arrived yet.

---

[360] *Id.* at 112.

[361] *Id.*

[362] *Id.* at 113

[363] For further detail on three-valued logic, *see* Ruth M. Kempson, *Semantic Theory* 139 (1977).

[364] Christopher Potts, *The Logic of Conventional Implicatures* (2005).

From this sentence, it may be inferred that the referent *she* is expected to arrive. This knowledge is associated with the semantic meaning of the word *yet* that enables the additional conveyed meaning. Conventional implicature is not dependent on context for its interpretation. This suggests that the problem with presupposition is that it distorts the distinction between semantic from pragmatics. A suggestion that has been raised by linguists is to differentiate instead between assertions and non-assertions, with non-assertions defined as the implicit knowledge or meaning that presupposed.[365] This arguably is a shift in nomenclature but does not tackle the issues at heart. That is, meaning is formed through a symbiosis of semantics and pragmatics. Ultimately, the discussion with pragmatics alludes to the indispensability of the subfield, particularly in maintaining the function of natural language. More importantly, the problems inherent to presupposition, and largely pragmatics, expose a fascinating parallel to the fact-law distinction. The next section transitions to computational linguistics and revisits the pillars of syntax and semantics.

Programming Languages: Technological Twin or Distant Cousin?

Discussions on linguistics frequently draw the analogy with computer programming. In particular, generative grammar and syntactic rules often are imagined as "command lines in a computer program."[366] Moreover, Chomsky's work was a fundamental source of inspiration for numerous theories in computer science.[367] Programming languages, as well, borrow linguistic terminology, expressing the construction and methods of interpretation through the lens of syntax and semantics. The following section aims to reflect on the similarities between programming and natural languages. Importantly, it tests whether key concepts[368] in programming are indeed functional equivalents to their linguistics siblings.

Programming languages generally evolved as a means of allowing machines to understand tasks. These languages, however, are not limited to the task of interpreting natural language. This suggests

[365] Birner, *supra* 324 at 120. *See* also Barbara Abbott, *Presuppositions and common ground*, 31 LINGUISTICS AND PHILOSOPHY 523 (2008).

[366] Carnie, *supra* 303 at 6.

[367] It is said that much of his early theory on formal languages became the basis of computational linguistics (such as the Chomsky Hierarchy). *See* Michael L. Scott, *Programming Language Pragmatics*, §2.4 (4th ed. 2016). *See* also the influence of Chomsky's *Cartesian linguistics*.

[368] I concede that I have cherrypicked some of the concepts for a more focused comparison with core linguistics. Namely, I highlight predominantly the perspective of language design and generation. While analysis is discussed, I do so in relation to the design. As well, I do not discuss the implementation of programming languages. Therefore, there are evident omissions in programming concepts. Again, it must be noted that the discussion is not intended to be exhaustive.

that while there may be programming languages for computational linguistics, the use of programming languages is not limited to processing language. As a result, computational linguistics may be a misnomer. The use of programming languages in the context of language may be applied more broadly. This will be discussed further in the section. It is important first to consider the fundamental building blocks of these languages. Just like natural language, programming languages follow constraints in expression and have an impact on how programmers can think.[369] Mirroring the order of this chapter, the section will start with syntax. Just as syntax in linguistics predicates on form, syntax in programming also bears a comparable connotation.[370]

Scanning and parsing are syntactic tasks in computer programming "to recognize the structure of a program without regard to its meaning."[371] A scanner reads a string of characters (i.e., consecutive series of natural language letters or numbers) and groups them into units (known as tokens). Interestingly, scanning is understood as a lexical analysis, with the primary purpose of simplifying the parsing exercise. Parsing, on the other hand, organizes the tokens into a parse tree. This assembly becomes a representation of the "higher-level constructs (statements, expressions, ... and so on),"[372] known as sequences. The overall structure then relies on a set of rules known as context-free grammar.[373] Context-free grammars are, therefore, considered the syntax of a programming language; the task of parsing belongs to the syntactic analysis. Consequently, any malformed tokens or unacceptable sequencing of them produces errors and syntactically invalid sequences.

Syntax centers on how structural rules are specified in a given programming language. It relies on regular expressions and context-free grammars. While syntax also enables those implementing programming languages to understand its structure, the intentions of the broader thesis focus on writing and analysis. As a result, how syntax is specified will be the primary point of discussion. The formal specification of syntax requires a set of rules.[374] There are four types of formal rules: (1) concatenation; (2) alternation; (3) "Kleene closure"; and (4) recursion. Concatenation is the joining of two or more-character strings. Alternation is the choice among a finite set of character strings.

[369] Scott, *supra* 367 at §1.2.

[370] *Id.* at §1.3.

[371] *Id.* at § 1.6.1.

[372] *Id.*

[373] *Id.*

[374] *Id.* at §2.1

Kleene closure is the repetition of character strings. Finally, recursion is the "creation of a construct from simpler instances of the same construct."[375] In other words, it is process of nesting.

A set of strings defined[376] using any of the first three rules becomes a regular language.[377] Regular languages are generated by regular expressions. Context-free languages (CFL), alternatively, are any sets of strings that are a combination of all four rules. CFLs are generated by context-free grammars. Regular expressions and context-free grammars are then language generators, specifying how to construct valid tokens or strings of characters. While regular expressions are able to define most tokens, they are unable to specify nested constructs.[378] It follows that the more complex the definition, the stronger the preference for context-free grammars. CFLs are then considered a superset of regular languages.

As discussed, syntactic structure may be revealed through parsing. Parsing deconstructs the grammar of a programming language and can be represented in a tree structure. When more than one parse (or syntax) tree can be constructed from a set of tokens, this is understood as ambiguous. Consequently, ambiguity falls under a similar understanding as the linguistic definition of "more than one." When ambiguity occurs, it signals that an additional mechanism must exist to "drive a choice between equally acceptable alternatives."[379] Some computer programmers work around ambiguity by including additional operators to eliminate multiple parses. This form of disambiguation is analogous with arithmetic calculations:

(1+2) * 5[380]

Relative to ambiguity is the notion of nondeterminacy. A nondeterministic construct, like ambiguity, is understood as having a choice between alternatives.[381] The difference, however, is that nondeterministic constructs are deliberately unspecified. That is, the particular options available are

[375] *Id.*

[376] As a clarification, the use of "defined" in a programming language is equivalent to the act of "writing" or "drafting" in natural language.

[377] Language is understood here not in the form of communication, but simply as a set of strings generated from the grammar. See *id.*

[378] *Id.* at §2.1.2.

[379] *Id.* at §2.1.3.

[380] For example, consistent with the arithmetic rules, the bracket signals that one must add first prior to multiplying.

[381] Scott, *supra* 367 at §6.7.

left to the decision of the user. This is fascinating as it implies that the choice among nondeterministic alternatives must be 'fair.'

Preliminary observations suggest that there is incredible overlap between syntax in core linguistics and syntax in programming. Both are highly rules-based and concerned with the structural construction of the language. More importantly, they consider the 'validity' of the grammar, specifying the points at which errors may be found in their expression. Likewise, syntactic considerations reflect on the structural relationships between entities. That is, both forms of syntax reflect on its potential for ambiguity. However, unlike syntax in core linguistics, the syntax of programming languages is not preoccupied with referencing and qualifying the identities of its components. Syntax analysis for programming languages is 'purely' structural.

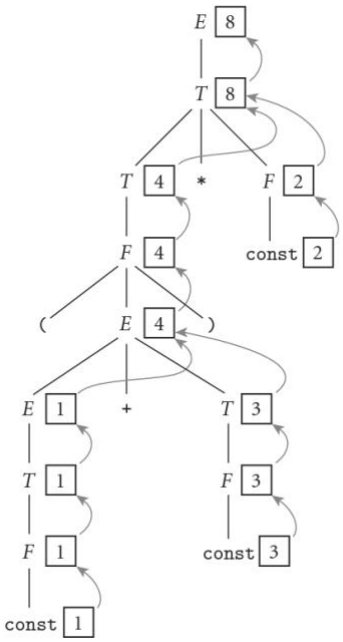
Semantic analysis, on the other hand, is "the discovery of meaning in a program."[382] A semantic function can recognize when multiple occurrences of the same token are intended to refer to the same entity. Equally, semantic analysis also identifies the types of expressions to ensure consistent usage and annotates them, such as verifying that entities are not used in an inappropriate context.[383] These annotations are known as attributes. Attributes are then described to 'decorate' syntax trees. It follows that attribute grammars provide a framework for the 'decoration.'[384] Below is an example of a syntax tree for (1 + 3) * 2:[385]

[382] *Id.* at §1.6.2.

[383] *Id.*

[384] *Id.* at §4.

[385] Figure 4.2. See *id.* at §4.2.



Simply, the attributes explain the structural interactions of the context-free grammar. Attribute grammars have two kinds of permissible rules: (1) copy rules; and (2) semantic functions. The former specifies that one attribute is a copy of another.[386] The latter specifies that one attribute is a product of an arithmetic operation. Below is an example of context-free grammar with its associated attribute grammar:[387]

1.	$E_1 \longrightarrow E_2$ + T	E_1.val := sum(E_2.val, T.val)
2.	$E_1 \longrightarrow E_2$ - T	E_1.val := difference(E_2.val, T.val)
3.	$E \longrightarrow T$	E.val := T.val
4.	$T_1 \longrightarrow T_2$ * F	T_1.val := product(T_2.val, F.val)
5.	$T_1 \longrightarrow T_2$ / F	T_1.val := quotient(T_2.val, F.val)
6.	$T \longrightarrow F$	T.val := F.val
7.	$F_1 \longrightarrow$ - F_2	F_1.val := additive_inverse(F_2.val)
8.	$F \longrightarrow$ (E)	F.val := E.val
9.	$F \longrightarrow$ const	F.val := const.val

[386] *Id.* at §4.2.

[387] Figure 4.1. See *id.* at §4.2.

Interestingly, the attribute grammar discussed appears to be akin to the notions of coindexing,[388] or more broadly, the labelling of structural relationships between constituents. In core linguistics, these are syntactic concepts. The semantic analysis of programming languages then seemingly embodies syntactic behaviors. Moreover, the terminology of "context" and "meaning" is used rather differently. Context and meaning in programming describe the act of qualifying an entity, as opposed to the process of deriving its substantive content. The syntax of programming languages is then analogous with defining the steps of a recipe; the semantics is equivalent to articulating the function of the ingredients. Both, however, do not express what the ingredients are and what would be achieved.

a. A Logical Intervention

The aforementioned descriptions of programming language syntax and semantics are broadly categorized as traditional imperative or prescriptive approaches. Herein enters the declarative or descriptive approach. Logic Programming is a style of declarative programming that applies the language of Symbolic Logic.[389] That is, logic programming relies on predicate and propositional logic in its operations. Logic programming is focused on defining "what is true and what is wanted."[390] It models sets of facts (known as datasets) and rules to define the "views of the facts in datasets."[391] Changes to facts are described as primitive updates.

Interestingly, logic programming is preoccupied with the conceptualization of worlds. It is concerned with defining in terms of objects and the relationship between objects. Objects are loosely understood as things, and relationships are the properties of the objects or relations among them. Objects are referred to as symbols and relations are predicates. A description of the relationship between symbols is understood as facts. Facts are frequently represented in a sentential form, consisting of the name of a relationship and the objects involved. See for example:[392]

```
parent (alex, megan)
```

[388] To recall, coindexing is the structural understanding of relationships between nouns or noun phrases.

[389] Michael Genesereth and Vinay K. Chaudhri, *Introduction to Logic Programming* 3 (2020).

[390] *Id.*

[391] *Id.* at 6.

[392] Derivative of example from Genesereth and Chaudhri, see *id* at 10.

Again, a set of facts form a dataset. Importantly, datasets are assumed to be true.[393] If a fact is not included in a dataset, it is presumably false. Logic programming acknowledges that more than one conceptualization is possible and suggests that "any conceptualization of the world is accommodated."[394] In short, what matters is utility and that within the world created, objects and their relations are expressed formally.

Rulesets, known as view relations, are interactions with the dataset. Applying the above example, with the knowledge of the relationship between Alex and Megan, can the grandparent relationship be "computed"? One method is to add facts to the dataset. This, however, is regarded as tedious. Alternatively, view relations can be established. See for example:[395]

```
grandparent (X, Z) :- parent (X, Y) & parent (Y, Z)
```

The above rule is indicative of the potential taxonomy and the establishment of hierarchies on the basis of facts and rules. While the syntax is less stringent[396] in logic programming, the semantics are rather important. The semantics of logic programming languages are the result of applying a set of view relations to a dataset, such that all conclusions rendered are true. This creates what is known as a closed logic program.[397] In effect, the semantics in programming languages are interpretable as logical entailment: conclusions must be true provided that all facts are true, and all facts required by the rules are true.[398]

Undoubtedly, concepts found in logic programming languages are reminiscent of those in core semantics. Namely, logic programming focuses on the creation of worlds and establishing the conditions of truth within these sets of worlds. More importantly, semantic meaning and logic programming are both fundamentally underpinned by predicate logic. Both 'linguistic' systems lean on logical operators. Therefore, it is perhaps owed to the similarities, between the semantics and

[393] *Id.* at 11.

[394] *Id.* at 18.

[395] *Id.* at 60.

[396] Syntactic restrictions do not raise errors or generate invalid formulation, only issues of compatibility. That is, ordering and structural interactions matter less than consistent use of the same symbols and predicates. See *id.* at 61.

[397] *Id*

[398] *Id.* at 63.

syntax of programming and natural language, that the application of computational linguistics to legal text appears as a logical next step.

Nevertheless, the 'linguistic' characteristics of programming languages are not exact parallels to natural language. There are subtle but substantive differences in their expression. Specifically, programming languages can either accord more closely with syntactic or semantic concepts in core linguistics, but not necessarily both (c.f. imperative with declarative programming). Moreover, there is no uniformity in the choice of programming language for computational linguistics. This suggests that there is potential variability in both the understanding and breakdown of text.

This is foreseeably problematic, as programming languages are fundamentally task-based. The intentions of their design are not to capture the nuances of natural language and meaning. Instead, they are built for versatility and are multifunctional. Consequently, the similarities between the semantics and syntax of programming and natural language are illusory. Even if the 'task' for the programming language, as in computational linguistics, is to understand language, it cannot completely to do so at this stage. It is my hypothesis that programming languages fail to account for a key pillar of natural language: pragmatics. Pragmatics has revealed that the communication of information relies, in part, on implicit knowledge. Currently, only explicit knowledge can be conveyed in programming.

Levelling the field: Reconciling Computation and Language

Equally, there must be a clarification between computational linguistics and computation *and* language. Computational linguistics uses programming languages to 'read' and 'interpret' existing texts written in natural language. It does not, however, contribute to the drafting of texts in code. This is misleading, as computational linguistics appear to be the standard for language treatment and is frequently referred to as the method of interpreting text. As a result, computational linguistics often falls within the field of natural language processing (NLP). This type of technology primarily relies on statistical probability and machine learning. In short, computational linguistics uses programming languages to approximate meaning.

Alternatively, computation *and* language use programming languages to create text and model linguistic behavior. Computation *and* language align with notions of knowledge representation. This is a far more complicated exercise that involves translating expert knowledge into a series of formal

structures understandable to machines.[399] Interestingly, the impression of similarity between the 'linguistic composition' of programming and natural language misconstrues the two forms of computationalism as two sides of the same coin. Computational linguistics has been helpful to the extent of performing high volume rapid review of texts. As computational linguistics rests on efficiency, it is largely preoccupied with rough approximations of word and sentence meaning. Accordingly, deeper analyses regarding the role of language and its relationship with meaning are not within the scope of its technological competence. I suggest that computation *and* language should be the path forward, particularly in the context of exploring the limits of legal expression.

Michael Reddy describes the metaphor that language is a conduit and words are containers.[400] Reddy suggests that content is considered as synonymous with thought, "ideas," and "meaning."[401] The words have "insides," such that thoughts may be inserted into them.[402] Consequently, communication is done by placing meaning into word containers, packaged neatly and transferred to the recipient to be unboxed. Consider the following sentences:[403]

> His words were hollow – he didn't mean them.
> Derrida's texts are rather deep.

Communication appears then to be a process of extraction. From the Conduit Metaphor, it may be inferred that words are merely one form of container for thought. So long as there is a place for meaning to reside, communication is possible. This could perhaps imply a "1-to-1 conversion" between natural and programming language. Could code be an alternative container?

However, as discussed in the prior chapter, the use of a particular language and the tools the language affords, impact and constrain thought. In linguistics, the infamous Sapir-Whorf Hypothesis stipulates that language affects conceptions of reality.[404] While there has been debate about the limits of this

[399] Harry Surden, *Artificial Intelligence and Law*, 35 GEORGIA STATE UNIVERSITY L. REV. 1316 (2019).

[400] Michael J. Reddy, "A case of frame conflict in our language" in A. Ortony (ed.) Metaphor and Thought 166-167 (2nd ed. 1993).

[401] *Id.* at 168.

[402] *Id.*

[403] Example derived from Reddy. See *id.*

[404] Benjamin Lee Whorf, *Language, Thought, and Reality* 134 (1956).

theory,[405] linguists generally acknowledge that language has an influence on thought. In accordance with this premise, would it not suggest that legal conceptions are already framed in natural language? Perhaps echoing Derrida, legal concepts cannot inherently be removed from its natural language encasing. The following chapter, thus, investigates the translation of legal texts from natural language to computer code. Through a series of case studies, I aim to question how programming languages have raised challenges around the computability of legal text and whether the law is indeed married to its language.

[405] There is spectrum around the 'strength' of this view: from linguistic determinism to linguistic relativity. The former suggests that reality is filtered by language. The latter is that thought is merely affected by language.

3- Case Studies on Translation*

* Earlier iterations of the case studies have either been published or are forthcoming in law journals and books alike, including notably the MIT Computational Law Report and the Northwestern Journal of Technology and Intellectual Property. Moreover, the second case study is drawn from an ongoing interdisciplinary research project of which I am an active member. The case studies presented here have been adapted to account for new findings and potential next steps.

3A- Writing in Sign (Computable Contracts)

Since the twelfth century, mathematical logicians allegedly used logical paradoxes to spot 'false' arguments in courts of law.[406] It was not, however, until the seventeenth century when Gottfried Leibniz proposed a mental alphabet;[407] whereby thoughts could be represented as combinations of symbols, and reasoning could be performed using statistical analysis. From Leibniz, George Boole's infamous treatise, *The Laws of Thought*, argued that algebra was a symbolic language capable of expression and construction of argument.[408] By the end of the twentieth century, mathematical equations were conceivably dialogic; a form of discourse.

This was perceivably owed to Boole's system; that complex thought could be reducible to the solution of equations. Nevertheless, the most fundamental contribution of Boole's work was the capacity to isolate notation from meaning.[409] That is, 'complexities' of the world would fall into the background as pure abstraction was brought to center stage. Eventually, Boole's work would form the basis of the modern-day algorithm and expression in formal language.

ASCII, the acronym for the American Standard Code for Information Interchange, is an exemplary case. Computers are only capable of understanding numbers. For a computer to interpret natural language, ASCII was developed to translate characters to numbers. Using a binary numeral system, ASCII assigns a numerical value – 32– to a letter. In brief, by performing the mathematical calculation, a binary code of 0s and 1s could be computed from a letter. Early conceptual computing devices, such as the Turing machine, were borne into existence as a direct product of Boolean algebra.

Christopher Markou and Simon Deakin point to the breakthroughs in natural language processing (NLP) as specifically contributing to the emergence of 'Legal Technology (Legal Tech).'[410] Markou and Deakin cite Noam Chomsky as inspiring early researchers of AI to design "hard-coded rules for capturing human knowledge."[411] Chomsky's work eventually contributed to powering advances in

---

[406] Keith Devlin, *Goodbye Descartes: The End of Logic and The Search for a New Cosmology of the Mind* 54 (1997).

[407] *Id* at 62.

[408] George Boole, *The Laws of Thought* Chapter 1 (1854).

[409] Devlin, *supra* 406 at 77.

[410] Christopher Markou and Simon Deakin, *Ex Machina Lex: The Limits of Legal Computability*, Working Paper (2019), available at SSRN: https://ssrn.com/abstract=3407856.

[411] *Id.* See also cited reference, E Brill and RJ Mooney, *Empirical Natural Language Processing*, 18 AI Magazine 4 (1997).

machine translation and language mapping. Known as expert systems, NLP applications "relied upon symbolic rules and templates using various grammatical and ontological constructs."[412] These achievements were then further enabled by Deep Learning[413] models, able to abstract and build representations of human language.

Computable contracts are making a powerful return. Contracts may be represented as computer data with terms made 'machine-readable' through a process of conversion: from descriptive natural language to consonant computer instruction. Conditions of agreements are not explained but listed as structured data records. Despite the capacity to express contracts in an alternative computable form, there is no means for interpretation. Instead, interpretation is perceived as irrelevant. Should digital data inscription and processing be considered a form of legal writing? If so, would it change the character of law?

The case study, therefore, follows the conundrum: what is the significance of the language in contract drafting? The project seeks to unpack several programming languages used in computable contracts. In identifying the logic of these languages, the project tackles methods of legal writing. The hypothesis is that, by analyzing the components of both legal and programming languages, a richer dialogue on the sociological implications of translating law to algorithmic form may be formed. Furthermore, it would be interesting to consider what contextual understanding may need to exist to 'interpret' contractual language.

The case study will unfold as follows. Part I will open with the current challenges and state of Legal Tech. Part II embarks on a brief investigation of programming languages, analyzing sample translations of contracts from natural language to computer code. Part III will gather early observations. Part IV will suggest implications for contract law and further considerations. Finally, I will conclude with a few remarks and possible next steps.

I. AS IT STANDS

[412] *Id.* at 11-15.

[413] Deep Learning is a subset of machine learning that involves artificial neural networks and the assigning of numerical weights on input variables. For further explanation, see *id* at 10-12.

Kingsley Martin spoke of the two greatest barriers to legal technology: (1) adjudication; and (2) language.[414] He teased at the subtlety and nuances of human communication. Meaning, he notes, could be changed with even the slightest adjustments to context. But beyond context, simple negations, "polysemy, synonymy, hyponymy and hypernymy"[415] are all functions of natural language that are obstacles for machines. He argues then that the general trend towards the simplification of language is rendering written legal documents, naturally, more machine-readable.[416]

Stephen Wolfram suggests that simplification could occur through the formulation of a symbolic discourse language. That is, if the "poetry" of natural language could be "crushed" out, one could arrive at a language that is entirely precise.[417] As opposed to translating meaning from natural language, the symbolic discourse language would be an alternative framing of the world. Could a distinct, symbolic representation of contractual language exist? What then are its implications?

Currently, expert systems and machine learning technology used for the revision of contracts seek to reduce the risk of human error. Eventually, contract analysis would manage, record, and standardize provisions that are 'proven favorable;'[418] in effect, perfecting contractual boilerplate. Boilerplate contracts are often regarded as a trade-off between tailoring and portability; that with broad standardization, the 'burden' of interpretation is lifted.[419] Contractual boilerplate, therefore, relies heavily on formalistic drafting, whereby form presides over meaning. For computable contracts, the migration of mediums – from descriptive natural language to mathematical form – generates data that identifies and signals the specific version of contracts that should be used in future cases.

---

[414] Kingsley Martin, "Legal Technology Barriers – Understanding Language and Exercising Judgment," Legal Executive Institute (September 24, 2015), https://www.legalexecutiveinstitute.com/legal-technology-barriers-understanding-language-and-exercising-judgement/.

[415] *Id.*

[416] *Id.*

[417] Stephen Wolfram, "Computational Law, Symbolic Discourse, and the AI Constitution," in Ed Walters (ed.), *Data-Driven Law: Data Analytics and New Legal Services* 152 (2019).

[418] Beverly Rich, "How AI is Changing Contracts," Harvard Business Review (February 12, 2018), https://hbr.org/2018/02/how-ai-is-changing-contracts. *See* also white paper "How Professional Services Are Using Kira," Kira Machine Learning Contract Analysis (accessed February 2019) available at: https://cdn2.hubspot.net/hubfs/465399/04-resources/whitepapers/KiraSystemsWhitePaper-HowProfessionalServicesFirmsAreUsingKira.pdf.

[419] Henry E. Smith, *Modularity in Contracts: Boilerplate and Information Flow*, 10 Mich. L. Rev. 1175, 1176 (2006).

A. Market Environment

Edilex, a Canadian Legal Tech start-up, is automating contract drafting by offering both AI-driven applications and downloadable legal document templates. Edilex's mission statement? The simplification of legal transactions and democratizing access to legal services. Genie AI is another fascinating Legal Tech start-up that offers AI-powered contract drafting. Using machine learning, the software recommends clauses to help legal practitioners "draft contracts faster."[420] Moreover, the technology marketed is focused on legal language, and one that is "suitable for lawyers."[421]

Evidently, the target demographic for each of the start-ups is rather different. The former is focused on the democratization of legal services; while the latter on enhancing the legal profession. Yet, both start-ups thrive on the notion of formalization; that there is a 'perfect' form achievable. By integrating AI in contract drafting, there is a push away from static mediums of writing. These include Microsoft Word (MS Word) and Adobe PDF; the original technological artifacts that evolved from pen and paper. In either case, the technology is never described as a replacement.[422] The purpose of these inventions is merely assistive.

B. Shifting Climates

Interestingly, the legal community is beginning to explore the problems associated with the use of static platforms like MS Word. Juro, for example, is a Legal Tech start-up that promotes contract management on a dynamic platform.[423] In a recent paper, Michael Jeffrey interrogates the use of MS Word as the dominant and default form for writing and editing legal documents. He considers the inefficiencies of manual updating, drafting, and reviewing. MS Word has been a prized product for legal drafting, Jeffrey notes. Though interpreted as a static platform, MS Word, in actuality, "can be

---

[420] "Super Drafter," Genie AI (accessed February 2020) https://genieai.co/home.

[421] See *id.* Genie equally advertises smart filters and an automatic knowledge base.

[422] Follows the existing literature that technology could only work complementary to the law. *See* Frank Pasquale, *A Rule of Persons, Not Machines: The Limits of Legal Automation*, 87 Geo. Wash. L. Rev. 2, 6 (2019). *See* also Neil M. Richards and William D. Smart, "How should the law think about robots?" in Ryan Calo et al, eds, *Robot Law* 16-18 (2018). Their chapter argues how law hinges on social and political relationships and metaphors that require a *latent* understanding of temporal social constructs (emphasis added).

[423] Based in London, Juro works platform translates contracts drafted in natural language to machine-readable form. Their platform allows contracts to be built in a text-based format that is also language independent (i.e. JSON). The contracts, thereby, exist in code. *See* Juro's whitepaper, Richard Mabey and Pavel Kovalevich, "Machine-readable contracts: a new paradigm for legal documentation," Juro Resources (accessed February 2020), available at: https://info.juro.com/machine-learning?hsCtaTracking=60e75e06-22bb-4980-a584-186124e645b3%7C6a7d3770-289d-4c97-bcfb-c9f47afec77f.

controlled through code."[424] In fact, MS Word has embedded in its software a number of templates modelled specifically for drafting legal documents. These templates contain automatic text entry, macros, and special formatting.[425] More recently, the startup Clause Logic, has developed an add-in that enhances MS Word's existing platform by automating clause creation and document assembly.[426]

Nevertheless, for long and complicated legal documents, Jeffrey argues that an integrated development environment (IDE) could "facilitate the authoring, compiling, and debugging" of contracts.[427] For programmers, the use of IDE provides several key features that are amenable to legal drafting. He notes the options for increased readability owed to color-coded syntax highlighting, automatic error detection, and predictive auto-complete features to provide suggestions while drafting. These features, he claims, could improve the drafting process by reducing the risk of human error and increasing efficiency.

Yet, the most interesting perspective he offers is the subtle equation of linguistic concepts as inherently mathematical.[428] Jeffrey draws programming concepts and applies them specifically to elements of legal drafting. The syntax, he notes, is "designed for drafting and document generation" and that the process would be "quite natural."[429] The underlying assumption is that the platforms of MS Word and an IDE have the same functional purpose. The differences lie in the added features for real-time changes. This speaks to a greater assertion: programming languages serve the same uses as natural language. But, the shift from pen and paper to MS Word did not fundamentally change the use of natural language for legal drafting. The use of IDEs, on the other hand, alters not only the platform, but also the method of execution.

Ultimately, the aforementioned start-ups, either Edilex or Genie AI, are only a few of the growing number of Legal Tech start-ups committed to the 'betterment' of contract drafting. These contracts

---

[424] Michael Jeffrey, *What Would an Integrated Development Environment for Law look like*?, MIT Computational Law Report Release 1.1 (2020), available at: https://law.mit.edu/pub/whatwouldanintegrateddevelopmentenvironmentforlawlooklike.

[425] "MS Word for Lawyers: Document Templates," Tech for Lawyering Competencies: Research & Writing (accessed May 2020), https://law-hawaii.libguides.com/TLC_Research_Writing/WordTemplates.

[426] "Our Technology," Clause Logic (accessed May 2020), https://www.clauselogic.com/.

[427] Jeffrey, *supra* 424.

[428] Jeffrey notes, "For legal drafting...the focus is linguistic – rather than mathematical – but the core concepts are the same." See *id.*

[429] *Id.*

are classified as more efficient, precise; otherwise, 'smarter.' There is, nonetheless, a dearth of literature on the use of formal languages for legal writing. Albeit, formal programming languages for contract drafting not only exist but have proliferated in the past few years. Their ancestors sprung from logic programming in the 1970s.

II. SO, CAN YOU CODE IT?

Even before the days of logic programming, contract drafting has seen symptoms of logic-based strategies in the literature since the 1950s. In "Symbolic Logic: A Razor-Edged Tool for Drafting and Interpreting Legal Documents," Layman E. Allen proposes the use of mathematical notation for the expression of contracts. He argues that its application will improve clarity, precision, and efficiency of analysis. He introduces six elementary logical connectives: implication, conjunction, co-implication, exclusive disjunction, inclusive disjunction and negation.[430] The most interesting connectives are implication and co-implication. These logical connectives are associated with the representation of causal relations; otherwise, "if X then Y" statements. Allen labels this form of expression as "systematically-pulverized"[431] and the process of transforming a statement to this form requires two primary actions: (1) divide statement into constituent elements; (2) and rearrange elements to approximate a 'systematically pulverized' form. Co-implication enhances the equation by including logical equivalencies. In sum, Allen teases at the age-old use of syllogisms in legal writing and provides an excellent backdrop to the study. In effect, how are programming languages applying logic to legal drafting?

Two of the most broadly used programming languages, Python and Prolog, use opposing methods of operation; the former is procedural, while the latter is declarative. Procedural programs often specify *how* the problem is to be solved. That is, with procedural programs, there are clear instructions for the program to follow. Akin to baking, all terms are defined explicitly, and all rules must be laid out. Should a program, such as Python, find that it cannot proceed with the task, this is typically because the program is unable to recognize the syntax. Equally, Python is incredibly sensitive to changes in the code; even a misplaced comma or indent in the spacing could affect the overall outcome of the specified task. Procedural programs often include functions; self-contained

[430] Layman E. Allen, *Symbolic Logic: A Razor-Edged Tool for Drafting and Interpreting Legal Documents*, 66 Yale L. J. 833 (1957).

[431] *Id* at 836.

modules of code capable of being manipulated and reused for innumerable tasks. Perhaps its most powerful operation, Python is able to examine and decide actions on the basis of conditions. Moreover, Python simplifies work by being able to loop through the same tasks in a given list. Rather than the manual repetition of a given task, Python is able to do so in a matter of seconds.

On the other hand, declarative programs specify *what* the problem is and ask the system, instead, to solve it. Declarative languages are founded on either the relationships (1) between objects; or (2) between objects and their properties. These relationships may be defined implicitly through rules or explicitly through facts. Facts describe relationships, while rules qualify them. The purpose of Prolog, therefore, is to form a fixed dataset that would derive answers to future queries about a relationship or set of relationships based on the inputted information. In contrast, the purpose of Python is to complete a particular task. While it can certainly account for prospective changes to the data, every step is explicitly expressed.[432]

Advancing forward several decades, Python and Prolog have become inspirations for a new era of programming languages used for drafting computable contracts. The project will explore a number of formal languages currently being prototyped. These include Ergo, Sophia, Solidity, Lexon, Blawx, and OpenLaw. While they certainly do not account for all the languages that are being workshopped, they are among the most broadly discussed in the Legal Tech sphere. Each language is built from different models. Ergo is a programming language modelled on execution logic for legal writing. It belongs to the suite of resources offered by The Accord Project.[433] Sophia and Solidity were both influenced by the Python syntax; created specifically for smart contract implementation.[434] Lexon and Blawx, on the other hand, are non-coding options with the former developed on declarative logic and the latter derived from linguistic modelling.[435]

---

[432] I acknowledge that Python is able to work in adaptive environments and does not have a fixed data set. The comment is directed at the explicit expression of a given task.

[433] The Accord Project also offers Cicero and Concerto. The former is a contract template generator that helps build agreements embedded with machine executable components. The latter is a program that enables the data of computable contracts to be manipulated and modelled. The Accord Project also offers a trial template editor to build and test out smart agreements. For more information, see "What is the Accord Project," Accord Project (accessed February 2020), https://www.accordproject.org/about.

[434] "The Sophia Language," Github Aeternity Docs (accessed April 2020) https://github.com/aeternity/aesophia/blob/lima/docs/sophia.md#stateful-functions. *See* also "Solidity," Solidity (accessed April 2020) https://solidity.readthedocs.io/en/v0.6.6/.

[435] Lexon qualifies its model as designed with the intention of reasoning in natural language and uses formal linguistic structure. *See* Henning Diedrich, *Lexon: Digital Contracts* (2020).

Finally, OpenLaw is more complicated to characterize. OpenLaw neither stems from Python nor Prolog. OpenLaw instead runs on Javascript[436] and uses a markup language to "transform natural language agreements into machine-readable objects with relevant variables and logic defined within in a given document."[437] These documents are then compiled together to act as contracts. Interestingly, the markup language allows for legal agreements to be enabled on the blockchain, but with natural language qualifiers.[438]

Prior to delving into the mechanics, there are a few disclaimers. First, I do not distinguish between machine-readable and machine-executable contracts. Rather than bifurcating the two architectural forms, the analysis focuses broadly on Smart Legal Contracts.[439] Next, to understand how formal languages may be used to draft contracts, I refer to extracts of legal documents translated from natural language to code. These translations are originals of each programming language, unedited and taken directly from their technical documentation. They were included as examples of how contracts may be drafted in the select language. The translations are, therefore, presumed to be manually done by each language's programmers; and thereby implicitly represent their design choices. As well, the formal languages analyzed are understandably evolving in their capacities. Consequently, the observations are only current to the time of this analysis. Finally, as there are, to date, no quantitative metrics to evaluate the existing pitfalls of contracts drafted in natural language. The study can only offer qualitative perspective on formal languages as a medium for legal drafting.

A. Ergo

To begin, Ergo follows a more traditional form of procedural programming and is largely function-based. This means that its language is predicated on the performance of the contract. However, Ergo is unique. It cannot be divorced from the overarching contract implementation mechanism, known as Cicero. Cicero consists of three 'layers': (1) text; (2) model; and (3) logic. Ergo is the logic

[436] Defined as a programming language with a code structure to build commands that perform actions. "Code Structure," The JavaScript Language (accessed April 2020), https://javascript.info/.

[437] "Markup Language," OpenLaw (accessed April 2020), https://docs.openlaw.io/markup-language/#variables.

[438] *Id.*

[439] I rely on the definition of Smart Legal Contracts as legal agreements that include digital components. These components allow the document to be interpreted and executed by computers. Both machine-readable and machine-executable contracts tie legal text to code.

component.[440] It is perhaps considered the 'end' process of a continuous flow of translation from human-readable to machine-executable.[441]

The Cicero architecture, therefore, is an interdependent network of resources that start with natural language text and end with compartmentalized data packages. That is, natural language contracts may be deconstructed into reproducible modules that can be interchangeably used between various types of contracts. How does this work?

Contractual clauses are sorted and categorized into qualitative and quantitative components. Descriptive terms of the contract remain at the text layer.[442] Variables that are quantifiable, on the other hand, are extracted from the natural language and captured in the model layer. These variables are notably bits of information that are reusable, iterative, and computable. This layer bounds natural language to data, as variables map conditions and relationships of the contract. Arriving at the logic layer, what remains are functional requirements of these variables. In other words, what are the specific operations necessary in order for these variables to perform the demands and terms of the contract?

Consequently, Ergo is intentionally limited with its expressiveness.[443] Consider the following contractual clause translated from descriptive natural language to Ergo.

The original provision, in prose, states:

> Additionally, the Equipment should have proper devices on it to record any shock during transportation as any instance of acceleration outside the bounds of -0.5g and 0.5g. Each shock shall reduce the Contract Price by $5.00.

The clause, in code, reads:

[440] "Key Concepts," Accord Project (accessed October 2020), https://docs.accordproject.org/docs/accordproject-concepts.

[441] *Id.*

[442] *Id.*

[443] The goal is for conditional and bounded iteration. This is presumably contributive to the reusability of contractual clauses. *See* "Ergo overview," Accord Project (accessed February 2020), https://docs.accordproject.org/docs/logic-ergo.html.

```
contract FragileGoods over FragileGoodsClause {
  clause fragilegoods(request : DeliveryUpdate) : PayOut emits PaymentObligation {
    let amount = contract.deliveryPrice.doubleValue;
    let currency = contract.deliveryPrice.currencyCode;
    let shocks =
      integerToDouble(count(
        foreach r in request.accelerometerReadings
        where r > contract.accelerationMax or r < contract.accelerationMin
        return r
    ));
    let amount = amount - shocks * contract.accelerationBreachPenalty.doubleValue;

    enforce request.status = ARRIVED and request.finishTime != none
    else return PayOut{
      amount: MonetaryAmount{
        doubleValue: amount,
        currencyCode: currency
      }
    };
```

Figure A Extracted from Ergo's 'Fragile Goods Logic,' (*Cicero Template Library,* Github) <https://github.com/accordproject/cicero-template-library/blob/master/src/fragile-goods/logic/logic.ergo> accessed October 2020.

At first glance, the translation is rather striking. There are evidently several omissions from the natural language text to the Ergo language. First, mention of recording devices that determine the weight changes are excluded from the code. Moreover, fluctuations in the Contract Price are equally excluded. Instead, only variables remain, such as DeliveryUpdate, PaymentObligation, accelerometerReadings, accelerationMin and etc.

Upon closer reading, it becomes clear that the contractual clause has undergone a decoupling process. That is, a conversion from the original unified contractual language to independent, actionable constituents has taken place. These variables are quantitative reconfigurations of the 'performative' elements of the contract. For example, the model layer reconstructs the weight changes and fluctuations in the Contract Price to:

```
},
"accelerationMin": -0.5,
"accelerationMax": 0.5,
"accelerationBreachPenalty": {
    "$class": "org.accordproject.money.MonetaryAmount",
    "doubleValue": 5,
    "currencyCode": "USD"
`
```

Figure B Extracted from 'Fragile Goods,' (*Accord Project*) <https://templates.accordproject.org/fragile-goods@0.14.0.html> accessed October 2020.

As noted, Ergo applies these variables and signals their operations. The Ergo language requests for the acceleration readings from the recording devices, then dependent on the parameter changes, computes whether the Contract Price would alter. This method of distilling the quantifiable from the qualifiable suggests that contracts are necessarily unambiguous and, in effect, are simply a matter of structuring.

B. Sophia and Solidity

Sophia is a language customized for smart contracts[444] on the Aeternity Blockchain.[445] The main unit of the code is focused on the performance of the contract. As the code is limited to contract implementation, the syntax of the language is again purely functional.[446] Prior to delving into the translation, it may be important to define a few key terms. First, the state is understood as the objects of the contract. The entrypoints are the actions pursuant to the contract. If the contract stipulates modifying the state, entrypoints are annotated with the 'stateful' keyword.[447] The inclusion of stateful is the dividing line between transactions and calls in smart contracts. The former requires modification; the latter does not. For example, a procurement contract requests a notice upon delivery. As the notice does not require modifying the state, a simple entrypoint would suffice. The actual delivery, on the other hand, would require the stateful qualifier. All in all, Sophia applies a Python-style syntactic structure with minor changes to the notation.

Consider the sample purchase agreement written in Sophia:

2. **BuyerContract** implements following functions:

- `deposit_to_seller_contract(price : int, key : address)` - the passing arguments are the price of the item and seller's contract address. There are functions implemented in Transport and Seller contracts, that we can call from Buyer contract, to check status of the item:
- `check_courier_status(transport_contract_address : address)` - will return the status of the order.
- `check_courier_location(transport_contract_address : address)` - will return the current location of the order.
- `check_courier_timestamp(transport_contract_address : address)` - will return the timestamp of last update.

[444] Smart contracts are defined in the paper as contracts limited to the enforcement of relationships through cryptographic code. *See* "How do Ethereum Smart Contracts Work?," Coindesk (March 30, 2017), https://www.coindesk.com/learn/ethereum-101/ethereum-smart-contracts-work.

[445] Defined as a scalable platform for executing smart contracts. *See* "Why aeternity is so innovative?," Aeternity (accessed April 2020), https://aeternity.com/.

[446] "The Sophia Language," *supra* 434.

[447] *Id.*

```
contract SellerInterface =
  entrypoint received_item : () => bool
  entrypoint seller_contract_balance : () => int
  entrypoint check_item_status : () => string

contract TransportInterface =
  entrypoint check_courier_status : () => string
  entrypoint check_courier_location : () => string
  entrypoint check_courier_timestamp : () => int

contract Buyer =
  stateful entrypoint deposit_to_seller_contract(price : int, key : address) : () =
    Chain.spend(key, price)

  entrypoint received_item(remote : SellerInterface) : bool =
    remote.received_item()

  entrypoint seller_contract_balance(remote : SellerInterface) : int =
    remote.seller_contract_balance()

  entrypoint check_item_status(remote : SellerInterface) : string =
    remote.check_item_status()

  entrypoint check_courier_status(remote : TransportInterface) : string =
    remote.check_courier_status()

  entrypoint check_courier_location(remote : TransportInterface) : string =
    remote.check_courier_location()

  entrypoint check_courier_timestamp(remote : TransportInterface) : int =
    remote.check_courier_timestamp()
```

The purchase agreement is remarkably direct. In the above contract, the terms of the agreement have been reduced to a mere 19 lines of code. The remainder of the agreement serves to notify delivery and updates on courier status. Notably, the contracts apply existing functions that have been pre-programmed; thereby, rendering performance automatic. Most purchase agreements are templates easily found with a quick search on the Internet. The programmed functions mirror the use of templates. Placeholders on templates are instead dynamic variables. Clauses that indicate qualitative expectations of the product for purchase (i.e. the condition of the good) remain as annotations outside of the contract.

Similarly, Solidity is another language used for the implementation of smart contracts. Solidity draws influence from Python and is an object-oriented language.[448] As opposed to the Aeternity Blockchain, Solidity is, instead, a language customized for the Ethereum Blockchain.[449] As opposed to states and

[448] "Solidity," *supra* 434.

[449] I have elected not to delve into the specifics of blockchain. This is simply to clarify that these languages, while similar, operate on different smart contracts platforms.

entrypoints, Solidity uses the syntax of variables and functions akin to 'Python-ese'. For example: rather than using 'stateful' as the performative, Solidity uses 'modifier.' Simply put, their uses parallel those of Sophia. Solidity, however, offers more options in qualifying contracting parties. Structs and Enums are syntactical operations that better classify the types of users engaged in the contract.[450]

Consider the sample purchase agreement written in Solidity.

```
contract Purchase {
    uint public value;
    address payable public seller;
    address payable public buyer;

    enum State { Created, Locked, Release, Inactive }
    // The state variable has a default value of the first member, `State.created`
    State public state;

    modifier condition(bool _condition) {
        require(_condition);
        _;
    }

    modifier onlyBuyer() {
        require(
            msg.sender == buyer,
            "Only buyer can call this."
        );
        _;
    }

    modifier onlySeller() {
        require(
            msg.sender == seller,
            "Only seller can call this."
        );
        _;
    }

    modifier inState(State _state) {
        require(
            state == _state,
            "Invalid state."
        );
        _;
    }
```

[450] "Structure of a Contract," Solidity (accessed April 2020), https://solidity.readthedocs.io/en/v0.6.7/structure-of-a-contract.html.

```
/// Confirm the purchase as buyer.
/// Transaction has to include `2 * value` ether.
/// The ether will be locked until confirmReceived
/// is called.
function confirmPurchase()
    public
    inState(State.Created)
    condition(msg.value == (2 * value))
    payable
{
    emit PurchaseConfirmed();
    buyer = msg.sender;
    state = State.Locked;
}
```

```
/// Confirm that you (the buyer) received the item.
/// This will release the locked ether.
function confirmReceived()
    public
    onlyBuyer
    inState(State.Locked)
{
    emit ItemReceived();
    // It is important to change the state first because
    // otherwise, the contracts called using `send` below
    // can call in again here.
    state = State.Release;

    buyer.transfer(value);
}
```

Again, the drafting of the purchase agreement is highly procedural and direct. There are no terms and conditions qualifying the object for purchase. Instead, there are only 'code-ified' limitations; measures to verify the identities of the contracting parties and confirm the purchase. All operations facilitate performance of the contract.

In both Sophia and Solidity, there are no translations of agreements from natural language to code. Rather, there are merely examples of contracts drafted in the formal language. That is, these contracts are reimagined in code at their creation. The translation process is internalized and configured to the parameters of the programming language. The purchase agreements 'speak the language'[451] of smart contracts. Certainly, for smart contracts, its uses extend beyond purchase agreements. Currently, the use cases for smart contracts are narrow and typically do not require qualitative accounts.[452] The issue perhaps is the conflation of other use cases with contracts in

[451] Recall Mireille Hildebrandt noting the shift to computation as one from reason to statistics. *See* Mireille Hildebrandt, "Law *as* computation in the era of artificial intelligence: Speaking law to the power of statistics," Draft for Special Issue U. Toronto L.J., 13 (2019).

[452] Smart contracts have been used for blockchain use cases such as the trading of cryptocurrencies, voting, or even blind auctions. *See* "Solidity by Example," Solidity (accessed April 2020), https://solidity.readthedocs.io/en/v0.6.7/solidity-by-example.html. *See* also Gideon Greenspan, "Why Many Smart

particular. For programmers well-versed in Solidity or Sophia, the identifiable problem is determining whether the purchased item had arrived at the buyer's address. *How* the good arrived is never the matter. By eliminating the how, there runs the risk of reducing contracts to a Boolean binary.

C. Lexon

Alternatively, Lexon is a peculiar mix to the programming languages studied. Unlike others, Lexon is founded on linguistic structure and designed to reason in natural language. Lexon reduces vocabulary and grammar to rule sets. Lexon's base vocabulary consists of definable 'names' used to designate objects and clauses. Just as one would draft sentences in natural language with a subject and predicate, Lexon operates in a similar fashion. There is, however, an important difference: articles are considered superfluous, 'filler,' words.

Below is a sample contract drafted in Lexon:

```
LEX Payment.

"Payer" is person.
"Payee" is person.
"Payment" is amount.

Payer pays Payment to Payee.
```

Articles (a, an, the) can be left out.

For an agreement at this level of simplicity, articles may not seem necessary to clarify the meaning of contractual terms. Nevertheless, party obligations do occasionally hinge on articles; potentially affecting the performance of the contract. It is not inconceivable that specifying a particular object as opposed to a general one matters, especially in certain procurement and sales contracts. Lexon

Contract Use Cases Are Simply Impossible," Coindesk (April 17, 2016), https://www.coindesk.com/three-smart-contract-misconceptions.

argues that the primary role of articles is to improve text readability. Yet, Lexon concedes that articles can "fundamentally change the meaning of a contract" and that this may be an area ripe for abuse.[453]

Further complicating the narrative, Lexon is not concerned about semantics altogether. The startup's creator, Henning Diedrich, acknowledges the inherent ambiguity of natural language that renders interpretation to be challenging; but argues that the Lexon language is not to clarify nor create complete contracts. Instead, Lexon is bridging the gap between formal programming and natural languages. Like other formal languages, Lexon cannot understand the 'meaning' of its terms. Its structural design only accounts for functionality. Lexon uses Context Free Grammars (CFG). First theorized by Chomsky, CFG do not depend on context; rules operate independent of the objects in question. Chomsky had originally developed CFG in an effort to formalize natural language. While this was largely unsuccessful in linguistics, it has since been popularized in computer science. Consequently, Lexon applies the model to create a programming language that is both expressible in natural language and readable by machines.

Diedrich contends that meaning could never be attained. Meaning is regarded as something that, though cannot be extracted, could be pointed to or described.[454] The Lexon language is structured in a manner reflective of these underlying assumptions. That is, rather than dwelling on the interpretation of the specific word or phrase in natural language, Lexon limits meaning to function. Diedrich states, "the actual functionality of the contract is the better description of ...the list of the actual rights and obligations of that person without relying on the original meaning of the word."[455] By framing functionality as a proxy for party obligations, Lexon inadvertently reframes the basis of contract theory from party autonomy to contract performance.

D. Blawx[456]

Blawx, on the other hand, uses a declarative logic. Perhaps the most interesting element of this language is its user interface. The code visually appears as puzzle pieces –or, Lego blocks – searching for its missing piece. Blawx was inspired by the program, Scratch, created in MIT as an educational

[453] Lexon has noted that future tools would account for the possibility such abuse. *See* Diedrich *supra* 435 at 33.

[454] *Id* at 107.

[455] *Id* at 106.

[456] It must be acknowledged that Blawx is currently in alpha version and at the early stages of a prototype. It has, however, been recognized for its potential as a legal reasoning and drafting tool.

assistant for children learning how to code. As the 'blocks' literally connect with one another, they visually capture the relationships between objects and their properties. Moreover, there is limited room for error; since the 'pieces' would physically not fit together should the code be written incorrectly.

Much like Prolog, Blawx operates on sets of facts and rules. Facts represent objects, or things, known to be true *in the code*. Rules are coded statements composed of both conditions and conclusions. Both elements are required in order for a rule to exist. Unlike other programming languages, Blawx works on the premise of declarative rules such that "conclusions are true if conditions are true." This may seem no different than traditional 'if, then' statements. This is surprisingly false. In programming, the 'if conditions then conclusions' framework operates temporally. For machines, this means that conditions only apply to the specific task at hand and do not apply globally to the program.[457] In the case of Blawx, rules are encoded in a declarative manner to help form the particular program's 'universe of knowledge.' Once the 'universe' of facts and rules have been established, the program will be able to answer to queries. Queries are fact-based and binary.

Blawx aims to transform legal documents to queryable databases. In practice, this would suggest that contracts may be encoded using the aforementioned logic of the program. Ultimately, the goal is for parties to be able to reason by simply asking binary questions to the application. The encoding of facts and rules allows parties to move from legal reasoning to legal information extraction. Interpretation, then, is no longer required since the solutions are presumed to be directly retrievable.

Consider the sample translation of a legislative act from descriptive natural language to Blawx. The article states:

> **5(1)**: A personal directive must
>
> (a) Be in writing,
> (b) Be dated
> (c) Be signed at the end
> i. By the maker in the presence of a witness, or

[457] This is described as "if right now the conditions are true, then next the computer should do conclusions." See "Facts, Rules, and Queries," Blawx.com (accessed February 2020), https://www.blawx.com/2019/09/facts-rules-and-queries/#page-content.

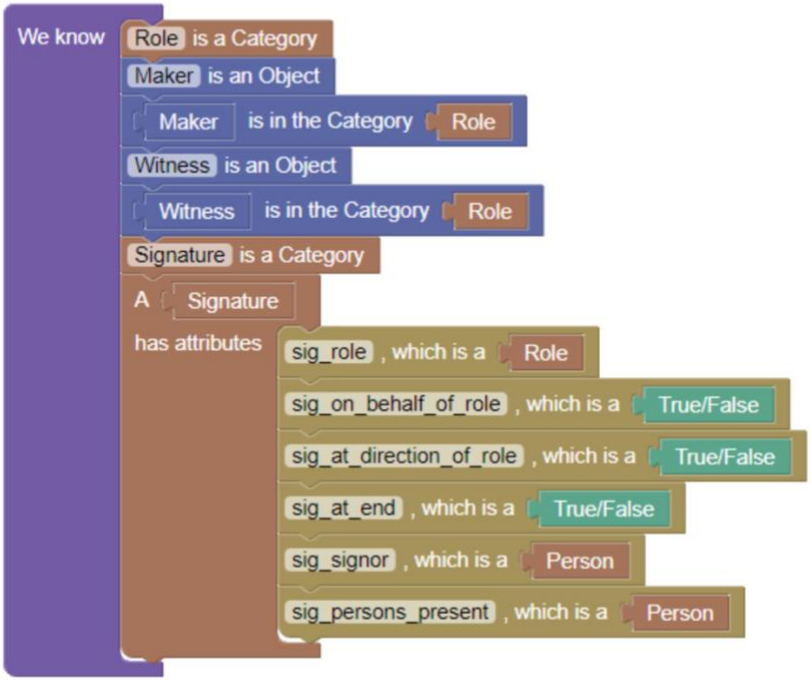
ii. If the maker is physically unable to sign the directive, by another person on behalf of the maker, at the maker's direction and in the presence of both the maker and a witness,

and

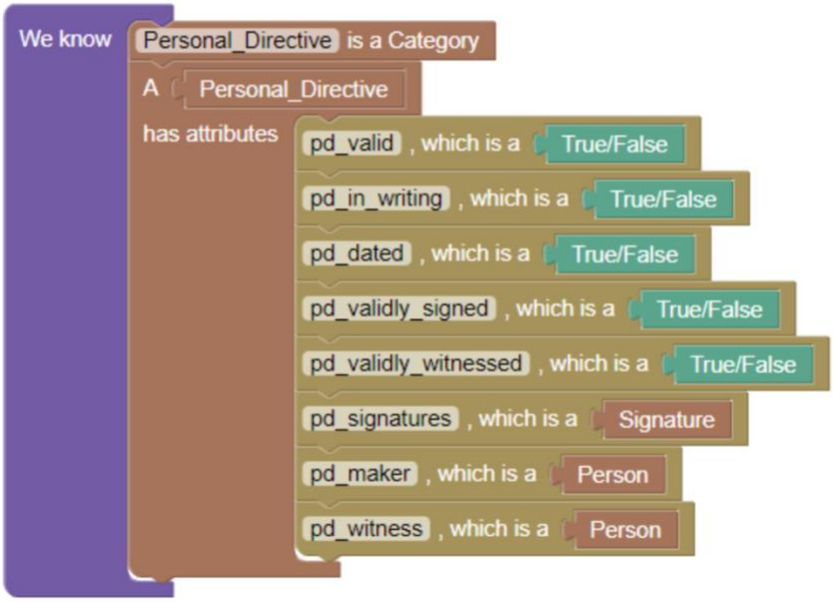
(d) Be signed by the witness referred to in clause (c) in the presence of the maker.

The provision, in Blawx, reads:

Here is what our new ontological elements look like. First, the Roles, and Signatures.



And this is the ontology for Personal Directives.



This translation is an especially difficult read. First, the 'block' appearance of the language may be troubling for those who are not tactile learners. The programming language forces the reader to focus instead on the conceptual components of the rules as opposed to the clause. The logic of the program necessitates a substantive breakdown of the legislation to its ontological elements. Simply put, it reduces the law to the relevant actors and their obligations. In this case, these elements are (1) the roles (actors); and (2) the signatures (obligations).

More importantly, the process of converting natural language to Blawx faced significant challenges with interpretation.[458] Coding the legislation required reframing the meaning of "personal directive"[459] into a binary; either as an object or an action. Fundamentally, it is a reconfiguration of the law to its function. Rather than, "what are the requirements of a personal directive," the question becomes "what actions must be taken in order for the personal directive to have legal effect?" The questions asked *de facto* bear the same meaning. The difference, while subtle, crucially points to an implicit recognition of the legal effect of the document in natural language. Notably, a personal directive could only exist should the requirements be met. Otherwise, it would simply be a piece of paper. This was raised as a note on the translation. Blawx introduced the concept of "validity" as a new condition[460] because there was no form of classification for a document that was not a personal directive. In the context of computable contracts, the Blawx language – like Ergo – would perhaps work best for contracts with clear objectives and unidirectional relationships.

E. OpenLaw

The last programming language perhaps poses as a stark contrast to the other formal languages studied.[461] For OpenLaw, the aim is not to translate the natural language agreements in their entirety. Instead, the language acts as a hybrid; an integration of machine-readable code with clauses drafted

[458] There is repeated commentary on the difficulty of interpretation when converting to a binary. "Example: Using Blawx for Rules as Code," Blawx.com (accessed February 2020), https://www.blawx.com/2020/01/example-using-blawx-for-rules-as-code/#page-content.

[459] Here, the personal directive is understood to be a 'living will.'

[460] Following the formula of a declarative rule, this would suggest "this is a personal directive (conclusion) if it is valid (condition)." Blawx, *supra* 457.

[461] I make the clarification here that the Accord Project also seeks to develop legal templates with associated computing logic. Nevertheless, while the Accord Project offers a similar form, the study focuses on the independent application of the Ergo language. *See* "Overview," Accord Project (accessed February 2020), https://docs.accordproject.org/docs/accordproject.html#what-is-a-smart-legal-contract.

in natural language.[462] The intention is to generate variables and logic to be imported and incorporated into forthcoming contracts of a specified type. For example, a non-disclosure agreement (NDA) typically would take the names of contractual parties and transform them as dynamic variables. If the variable requires further description, additional string[463] text could be used to qualify the term. Boolean logic is a feature of OpenLaw's programming language. The function, "conditionals," embeds logic in a legal agreement; reconstructing contractual terms into binary questions. Clauses are interpreted as "embedded template[s]."[464] The goal is to reduce drafting work by storing boilerplate clauses as data that may be added to contracts.

Below is an excerpt of an advisor agreement written in OpenLaw:

```
\centered **Simple Advisor Agreement**

This Advisor Agreement is entered into between [[Company Name]] ("Corporation")
and [[Advisor Name]] ("Advisor") as of [[Effective Date: Date]] ("Effective Date
Company and Advisor agree as follows:

^**Services.** Advisor agrees to consult with and advise Company from time to ti
at Company's request (the "Services").

^[[Choice of Law Insert: Clause("Choice of Law and Venue Clause")]]

^**Termination.** Either party may terminate this Agreement at any time, for any
reason, by giving the other notice.
```

[462] *See* "Markup Language," *supra* 437.

[463] In computer programming, a string is defined as a sequence of characters and is representative of text. *See* "String," TechTerms (accessed February 2020), https://techterms.com/definition/string.

[464] "Markup Language," *supra* 437

The excerpt of the agreement is presented in two forms: (1) in code; and (2) in OpenLaw's drafting editor. In either arrangement, the natural and formal language are woven together seamlessly. At first glance, it may be difficult to determine whether a translation exists. The enduring presence of the natural language and the structural consistency of the contract suggest the integrity of the agreement remains intact. Yet, the incorporation of code with natural language offers a dynamic interpretation of legal agreements. It mirrors the notion that select contractual elements are reproducible and calculable, while others require human intervention. The drafting process, however, is left rather unchanged. The hybrid approach is regarded as a method of simplification; identifying portions of the agreement that are quantifiable. The question becomes: what are the risks of simplification? Is 'hybridization' also translation?

In examining the programming languages, the technology is observably limited. Namely, contracts drafted in these languages are governing simple transactions. Nonetheless, they expose conflicting interpretations of contract theory. More specifically, a commonality across all formal languages is the interpretation of contracts as predicated on performance. Consequently, all languages are largely function-based. The principle of party autonomy, expressed often as details in contract terms, is only secondary to the actual completion of the transaction. Rather than what parties have agreed to and *how* the parties have fulfilled their obligations, it becomes solely dependent on *whether* the obligation has been completed. Negotiated contracts represent a 'meeting of the minds.' With

program languages, there runs the risk of reconfiguring basic contracts doctrines; conflating the principles of consideration as offer and acceptance as obligation. The exception, of course, is OpenLaw. Its hybrid approach raises provocative questions on the use of embedded code in legal drafting.

III. OBSERVATIONS AND ANALYSIS

With the increasing normalization of smart contracts, computer code could foreseeably become a vehicle in which contracts are drafted. The question remains: should programming languages be recognized as a form of legal language? The following section will analyze the observations taken from the study against existing literature. As discussed, function becomes paramount to computable contracts. Formal programming languages reveal that because natural language is indeterminate, a migration away from semantics to syntax could resolve the challenges relevant to interpretation.

This was the impetus behind the innovative start-up – also, cleverly named – Legalese. L4, their marketed programming language, is a domain-specific language (DSL) designed to "capture the particularities of law, its semantics, deontics, and logic."[465] Unlike other formal languages, their 'logic' draws influence from Prolog, but has been developed for the sole intention of expressing law.[466] The purpose of L4 extends beyond the general application of programming languages to legal language. Rather, L4 produces formally verified 'smart' contracts that equally could be transformed into PDFs written in natural language. The idea is that the 'legalese' of contractual terms is a seamless translation between code and natural language. Legalese co-founder, Alexis Chun, states, "legal as a utility, not a consultation."[467] This may well be the mission statements of the other programming languages.

The idea of L4 sprung from a programmer seeking to 'decipher' an investment contract written in 'legalese.'[468]

[465] "What is Legalese?," About our Company (accessed April 2020), https://legalese.com/aboutus.html#innovation-premise.

[466] *Id.*

[467] "AL Interview: Software is Eating Law – Legalese.com," Artificial Lawyer (July 29, 2016), https://www.artificiallawyer.com/2016/07/29/al-interview-software-is-eating-law-legalese-com/.

[468] "Why Computational Law?," Legalese (accessed April 2020), https://legalese.com/computational-law.html.

1.2.1 If the investment for the purpose of the Series B Funding is valued at not more than $32.5 Million, then the investors in the Note shall be entitled to convert the Note into Shares at a fixed valuation of $27.5 million.

1.2.2 If the investment for the purpose of the Series B Funding is valued at less than $40 million but not below $32.5 million, investors in the Convertible Note will be entitled to convert the Note into Shares at a 15% discount over the valuation of the Series B Funding (for instance, if the series B Funding is at a valuation of $35 million, then the investors in the Note shall be entitled to convert at a valuation of 35M less 15% discount);

1.2.3 If the investment for the purpose of the Series B Funding is valued at not less than $40 million but less than $47.06 million, investors in the Convertible Note will be entitled to convert the Note into Shares at a 15% discount over the valuation of the Series B Funding (for instance, if the series B Funding is at a valuation of $47.06 million, then the investors in the Note shall be entitled to convert at a pre-money valuation of 40M i.e. $47.06 million less 15% discount);

1.2.4 If the investment for the purpose of the Series B Funding is valued at not less than $47.06 million but less than $80 million, investors in the Convertible Note will be entitled to convert the Note into Shares at a fixed pre-money valuation of $40 million;

1.2.5 If the investment for the purpose of the Series B Funding is valued at not less than $80M but less than $100M, investors in the Convertible Note will be entitled to convert the Note into Shares at a fixed pre-money valuation of $45M; and

1.2.6 If the investment for the purpose of the Series B Funding is valued at not less than $100 million, investors in the Convertible Note will be entitled to convert the Note into Shares at a fixed pre-money valuation of $50 million.

The programmer then drafted a translation of the investment contract. It read as follows:

```
if( seriesB < 32.5 ) { conversion = 27.5 }
else if( seriesB < 40 ) { conversion = seriesB * 0.85 }
else if( seriesB < 47.06 ) { conversion = seriesB * 0.85 }
else if( seriesB < 80 ) { conversion = 40 }
else if(seriesB < 100 ) { conversion = 45 }
else { conversion = 50 }
```

Evidently, the translation takes from a specific excerpt of the contract; in particular, one that is markedly quantifiable. Nevertheless, what the translation highlights is the monotony of certain contractual clauses. Every provision follows a similar phrasal structure. In effect, the programmer is pointing to the innate formalism that exists in select legal language. Though drafted in natural language, the repetition of noun phrases in the aforementioned excerpt divorces context knowledge from interpretation. The result? The ability to distil and transform natural language to clear computable form.

A. Early Inspirations

In another fascinating analysis, Layman E. Allen reflects on ambiguity in legal writing owed to syntactic uncertainties. Allen considers alternative structural constructions to manage issues of

'between sentence' logic found in legal drafting.[469] He first engages in an exercise to deconstruct an American patent statute and notices immediately a difficulty with the word 'unless.' He asks whether the inclusion of 'unless' asserts a unidirectional condition or a bidirectional condition.[470] That is, does the clause mean (a) if not *x* then *y*; or (b) if not *x* then *y* and if *x* then not *y*?

Though nuanced, Allen exposes an ambiguity that muddies the legal force of the statute. An interpretation of 'unless' as a bidirectional condition raises the question of what "not *y*" would mean. In this particular case, this could affect whether exceptions are possible in determining eligibility for a patent. He later acknowledges that the sections of the statute immediately preceding and following provide sufficient context. Nevertheless, he maintains that language must have a clear structure. Though conceding that semantic uncertainties are often deliberate, structural uncertainties are often inadvertent.[471] Drawing inspiration from computer science, Allen argues that drafting requires replacing the use of imprecise terms (i.e. 'unless') and, instead, constructing sentences that use "lowest common denominators of structural discourse."[472] These include 'and,' 'or,' 'not,' 'if...then,' and 'if and only if...then.' The similarities with formal language are stark, begging the question: how does reducing language to its 'lowest common denominators' affect the complexity and richness of legal language?

In "Self-Driving Contracts," Casey and Niblett consider the gaps in contract theory owed to the ambiguity of natural language. They argue that, currently, natural language as a medium of legal expression allows contracts to be both intentionally and unintentionally incomplete.[473] Intentional incompleteness is interesting because it implies that general language circumvents the *ex ante* costs of decision-making and creates a space for changes in conditions. This, however, often leads to issues of enforceability; such as disputes about the definitions of "reasonable" and "material."[474] Consequently, 'self-driving contracts' would use machine learning algorithms and expert systems to remove questions of enforceability.

---

[469] Layman E. Allen, "Language, Law, and Logic: Plain Legal Drafting for the Electronic Age," B. Niblett (ed.) *Computer Science and Law* 76 (1980).

[470] *Id* at 77.

[471] *Id* at 96.

[472] *Id.* at 99.

[473] Anthony J. Casey and Anthony Niblett, *Self-Driving Contracts*, 43 J. OF CORP. LAW. 101, 112-117 (2017).

[474] *Id* at 113.

Much like 'self-driving' contracts, the aforementioned programming languages help automate the processes of contract creation and interpretation. As observed in the study, interpretation is internalized by the technical bounds of the programming language, as contractual clauses are constructed to reason purposively.

B. Ergo

For Ergo, the question remains whether contractual ambiguities are a mere consequence of improper structural representation. Notably, the migration from text-to-model layer implies the potential for mathematical precision from inception. Duncan Kennedy argues that, whether Hart or Kelsen, determinacy is a matter of degree.[475] Though legal drafting may be simplified through the act of sorting, assessing whether a clause is sufficiently amenable to reusability is a difficult ask. The underlying assumption for the Cicero architecture is that the simplification process will not eventually alter the method of drafting. Perhaps a better question: is there value to qualitative descriptive clauses in legal writing? That is, would the 'text' layer remain relevant going forward; and what is the significance of retaining the natural language component of contract drafting?

As discussed by Casey and Niblett, contracts are deliberately incomplete. Again, this is because contracts are manifestations of party intent.[476] In effect, *how* contracts are written frame the behavior of parties, and thereby influence its performance. Contracts that are negotiated tend to be less specific and have more room for interpretation. Performance is less likely to be exact. Yet, performance is not compromised despite the 'incompleteness' of the contract. Instead, the contract's incompleteness signals trust between parties.[477]

For Sophia and Solidity,[478] the translated clause removes specifications. Solidity and Sophia reconceptualizes the clause by broadening the scope of the obligation; reclassifying specifications from conditions to warranties. Effectively, Sophia and Solidity fixes the meaning of contractual terms and renders interpretation irrelevant. In the ordinary negotiations of a contract drafted in natural

---

[475] Duncan Kennedy, *Legal Reasoning: Collected Essays* 154 (Davies Group Publishers, 2008).

[476] Zev J. Eigen, *Empirical Studies of Contract*, Faculty Working Paper 204 (2012), available at: https://scholarlycommons.law.northwestern.edu/cgi/viewcontent.cgi?article=1203&context=facultyworkingpapers.

[477] *Id* at 17. Eigen references the study by Chou, Halevy and Murninghan. *See* Eileen Y. Chou et. al, (2011) *The Relational Costs of Complete Contracts*, IACM 24th Annual Conference Paper, available at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1872569.

[478] As Solidity and Sophia all raise similar challenges, the observations found are discussed collectively.

language, a dispute may arise over mutual assent and performance; perhaps whether parties have agreed to the finer details of the contract.[479] With these programming languages, mutual assent is automatic and indisputable. Perhaps illustrative of the design, Solidity or Sophia contracts only address "consideration, mutuality of obligation, competency and capacity."[480] Offer and acceptance are assumed. What becomes problematic is, again, the reconceptualization of consideration.

Contracts, then, call for ambiguity, and specifically semantic ambiguity. In isolation, programming languages like Ergo create the illusion that mutual assent is automatic and indisputable. Semantic ambiguities no longer exist, as contractual negotiations are limited to operations with little care for parties' preferences. This could potentially invoke a behavioral change since contracts would become primarily functional in nature. Equally, this could conceivably lead to a simplification of contracts and a convergence towards contractual boilerplate. But, just as Cicero operates through the trifecta of text-model-data, natural language is indispensable from contract drafting. The role of natural language becomes monumental, ensuring that the elements of trust and party autonomy are not compromised and, rather, maintain the heart of contracts doctrine.

C. Lexon

Lexon's language poses a similar puzzle. Readable in natural language, Lexon's verbs are coded such that they coincide with the performance of the transaction. Diedrich's formulation of meaning finds parallels with Ludwig Wittgenstein's writings. Wittgenstein argues that language, as used presently, extends beyond names and "dry dictionary entries with their definitions."[481] The actions derived from words are effectively married to their meanings. It is conceivable then that language could be no more than a list of orders and classifications. It follows that in abiding by the rules of association is

[479] One could consider *Chartbrook Ltd. v. Persimmon Homes Ltd.* [2009] UKHL 38, the infamous English contracts case on the interpretation of contractual terms. The dispute concerned the sum Persimmon Homes was contractually obliged to pay Chartbrook. The Court of Appeals ruled that the natural meaning of the language fell closer in line with Chartbrook's interpretation. This case is a fascinating example regarding the express intention of parties. Upon appeal, the House of Lords unanimously ruled in favor of Persimmon Homes, citing that Chartbrook's interpretation of the clause did not make sense in a commercial sense. Although the Court ruled on the basis of meaning, there was nevertheless comment on negotiations preceding contract formation could be cited as evidence of meaning.

[480] "Declarations," Accord Project (accessed February 2020), https://docs.accordproject.org/docs/logic-decl.html.

[481] Sheila Jasanoff, *Can Science Make Sense of Life?* 117 (2019). Wittgenstein considered language as a form of life; and thereby, linguistic expression is constructive of its being. *See* also Ludwig Wittgenstein, *Philosophical Investigations* 19 (1958).

to accept the inherent authority of its practice. Meaning is found in the performance of the word, and not in the understanding of it.

Lexon claims that it neither translates nor transforms thought.[482] Instead, Lexon preserves the natural language construction of 'meaning,' by placing a constraint on its rules. That is, Lexon uses a subset of natural language grammar as the programming language of the legal contract.[483] This approach is known as "controlled natural language." Rather than processing *all* of natural language, a machine need only to process an assigned vocabulary and grammar. The assigned set becomes the operatives of the language game. Equally, Lexon wears the legacy of Chomskyan formal semantics; whereby the syntactic structure is both a projection and vessel of its function. Interpretation is again internalized by "mapping...symbols to a reference structure."[484]

D. Blawx

Blawx, alternatively, required defining in advance the actions of contractual parties. Again, the code internalizes interpretation as a preliminary step. Using a declarative logic, Blawx must first set the parameters of its dataset. On several occasions,[485] the code required defining new categories and forming different classifications in order to be amenable to translation. This involved making explicit the relationship between legal objects and their properties. Interestingly, legal questions, particularly those assumed to be accommodating to mathematical configuration, were found to be challenging in the Blawx language. For example, the determination of a personal directive could easily be structured as a binary question. Still, it was necessary to define the object that did not fulfil the requirements of a personal directive. This subsequently provoked a deeper question on the implicit recognition of legal documents.

Simply put, Blawx exposed the tacit force of law. Reflecting on H.L.A. Hart, the underlying assumption of "power-conferring rules [...] exist not in virtue of some further law-making act, but in virtue of a fundamental rule of recognition implicit in the practice of law-applying officials."[486]

---

[482] Diedrich, *supra* 435 at 104.

[483] *Id.*

[484] Giosuè Baggio, *Meaning in the Brain* 62 (2018).

[485] Blawx had encountered difficulty with interpreting the natural language of the legislation. Blawx recognized that it took 'creative liberties' in converting the statute to Blawx language. *See* Blawx, *supra* 458.

[486] H.L.A. Hart, *The Concept of Law* Chapters 4, 6 (1961).

Similarly, J.L. Austin contemplated the performative effect of 'utterances.' Austin uses the act of marriage to demonstrate how the utterance of a certain few words puts into effect its meaning.[487] Austin suggests that legal and moral obligations are relative to public specification; that utterances necessarily correspond with particular procedures situated within social contexts. Their mis-performance leads to a nullification or voidance of the act.[488]

In the case of Blawx, the meaninglessness and inability to articulate the 'inverse' of a legal document (i.e. missing the signature of a witness but would otherwise be a personal directive) points to the implicit dimension of the law.[489] The dividing line between a document having legal force, or not, speaks to the inherent authority of legal rules. Just as marriage could only be recognized within a specific circumstance, it was necessary for Blawx to acknowledge the deeper context; that is, "how is legal recognition being defined?" Blawx then applied a purposive interpretation, classifying legal recognition as validity. While the translation is rather sound – and validity is often a proxy for determining legal effect – the questions asked are distinct. From "is it legal" to "is it valid" is necessarily distinguishable in contract law. A contract may be valid but legally unenforceable. Therefore, interpreting legal force as validity subverts existing contract theory and, again, narrows interpretation to seemingly functional equivalents. Casey and Niblett are correct in noting that there will be an attempt to "pigeonhole [computable contracts] into existing frameworks of thought."[490] For Blawx, its uptake would likely require changes to existing contracts doctrines.

The challenge of using programming languages centers on interpretation. Drafting contracts in formal programming languages highlights the ambiguity of the original source. The task of translating contracts from descriptive natural language to code brings to light underlying assumptions of legal authority and re-evaluates party autonomy in contract theory. In nearly all the cases, the interpretative

[487] John L. Austin, *How to Do Things with Words* 7 (1975).

[488] *Id* at 16.

[489] Gerald J. Postema, *Implicit Law*,13 Law and Philosophy 361 (1994). Recall also, Allen and the difficulty of interpreting what is "not y" *See* Allen, *supra* 469. There is an alternative argument that Blawx may not be the right choice in programming language for particular types of law (i.e., legislation). That is, procedural languages could perceivably be a better option. Python, a procedural language, could construct a personal directive on the basis that the requirements are fundamentally conditional. There may be merit to a deeper investigation as to whether certain programming languages are more conducive to specific types of contracts.

[490] Casey and Niblett, *supra* 473.

exercise was done *ex ante*; that the contract's legal effect was established in direct parallel to performance.

IV. IMPLICATIONS AND FURTHER CONSIDERATIONS

As mentioned, formal programming languages have the impact of unifying legal concepts such as mutual assent with performance; effectively, reinvigorating arguments associated with contractual boilerplate.[491] Alternatively, it raises an argument for increased granularity by breaking down and identifying the conceptual components of contracts to specific executable tasks programmable in the language. In either case, there is a definite reframing of contracts doctrines. Derrida comes to mind: is the use of computer code for legal writing beyond 'convenient abbreviation'?[492] Hofstadter would argue for the case that computer code cannot be devoid of meaning and would indeed imprint its effect to the system. Hofstadter states, "[w]hen a system of 'meaningless' symbols has patterns in it that accurately track, or mirror, various phenomena in the world, then that tracking, or mirroring imbues the symbols with some degree of meaning..."[493] Structure cannot be divorced from meaning.

Recall Duncan Kennedy tested the relationship between structure, or symbols, and meaning by deconstructing argument into a system of 'argument-bites.' Argument-bites form the basic unit and such bites often appear in opposed pairs. Operations then performed on argument-bites constitute and build legal arguments. Such operations diagnose and assume the circumstances, or relationships, in which the argument-bite is to be manipulated and 'deployed.'[494] Such import of structural linguistics conceptualizes law and argument as systematically formulaic; "a product of the logic of operations."[495] Perhaps most interesting about Kennedy's theory is his idea of 'nesting.' Kennedy describes nesting as the act of 'reproduction' or the "reappearance of [argument-bites] when we have

---

[491] Boilerplate contracts as lifting the burden of interpretation and ensuring enforcement. Computable law borrows and extends the characteristics of contractual boilerplate in the name of increased precision, efficiency, and certainty. *Recall* Smith, *supra* 419.

[492] Jacques Derrida questioned natural language and the medium of writing as the accepted form of communication. His argument strikes an interesting parallel to the use of written and descriptive language in law. Derrida considers how writing is perceived as the original form of technology; that "the history of writing will conform to a law of mechanical economy." Writing was a means to conserve time and space and was independent of structure and meaning. *See* Jacques Derrida, *Limited Inc.* 4 (1988).

[493] Douglas Hofstadter, *Gödel, Escher, Bach* preface-3 (Twentieth-anniversary ed. 1999).

[494] Kennedy describes relating argument-bites to one another by such operations as a means of confronting legal problems. *See* Duncan Kennedy, *A Semiotics of Legal Argument*, 3 Collected Courses of the Academy of European Law 317, 351 (1994).

[495] *Id* at 343.

to resolve gaps, conflicts or ambiguities that emerge [from]...our initial solution to the doctrinal problem."[496] Therefore, the conundrum surfaces where language may be applied to law in a mechanical fashion but the process of reducing legal argument to a system of operations raises considerations on the act of labelling and the power in its performativity. That is – and as Kennedy rightfully notes – "language seems to be 'speaking the subject,' rather than the reverse."[497]

Kennedy's thought exercise is precisely analogous to the use of formal programming languages for legal drafting. Perhaps the question asked is not whether programming languages *should* be a legal language, but *how* they could be amenable to the demands of contract law. Are these demands to create more complete contracts, or to limit ambiguity and ensure contract enforcement? Thus far, the paper has sought to raise a number of concerns relevant to the use of programming languages, particularly in the translation of contracts from natural language to code. These concerns speak to whether the effort to complete contracts or disambiguate contractual terms could resolve inherent tensions of contract interpretation and enforceability.

A. The Spectrum

Modularity theory for the design of contracts has made a triumphant return in recent scholarship.[498] Recalling Smith, "natural language comes in varieties that are more or less formal."[499] As seen in Legalese's example of the investment contract, there are undeniably contractual clauses that are more formalistic than others. The trade-off, Smith claims, between context-dependence and formalism relies on the "amount of information conveyed"[500] within a particular provision. The amenability of the clause to reach a larger audience and wider variety of situations – thereby, more information-intensive – determines the degree of formalism applicable.[501] In other words, genericism mirrors formalism.

---

[496] *Id* at 346.

[497] *Id* at 350.

[498] *See* Smith, *supra* 419. *See* also George Triantis, *Improving Contract Quality: Modularity, Technology, and Innovation in Contract Design*, Stanford Law and Economics Olin Working Paper No. 450 (2013); and Matthew Jennejohn, *The Architecture of Contract Innovation*, 59 B.C.L. Rev 71 (2018).

[499] Smith, *id* at 1204.

[500] *Id.*

[501] *Id.* at 1206.

This appears to be the approach taken by OpenLaw. The method of integrating code with natural language suggests that a latent assessment of formalism should be applied to contracts. In the sample advisor agreement, the Choice of Law and Venue clause was determined to be highly reproducible. Leaning on Smith, the provision was likely to be rather broad and contained language generic to most advisor agreements. The clause would, therefore, satisfy the test; that it is amenable to translation. The OpenLaw method has seen adoption by the legal industry. Perhaps acknowledging the limitations of contracts drafted entirely in a programming language,[502] King and Wood Mallesons (KWM) have piloted a hybrid 'architecture' that combines "computational code and human discretion to produce a single contract[...]"[503] In the "ordinary lifecycle of [a] contract" where there is "nothing unpredictable,"[504] performance reigns supreme and that such performance could be easily automated. Yet, complexity in the market renders prediction impossible; that human judgment is required to assess the "extraordinary range of possibilities...facts which are far beyond the scope of any contract."[505] The solution, KWM recommends, is a 'seamless bond' between terms drafted in computational and natural language. The contract should be designed together to avoid the risk of "complicating the legal framework through inconsistent terms."[506]

KWM's project is remarkable and touches on the significance of legal design. The question then becomes one of operation. Using programming languages to draft contracts could pose challenges akin to incorporating contractual boilerplate to new contracts. As Richard Posner argues, clauses "transposed to a new context may make an imperfect fit with the other clauses in the contract [...]"[507] KWM seeks to overcome Posner's objection by actively acknowledging the significance of the legal relationship at the heart of contract law. But, drafting in tandem contractual clauses in code and natural language is a difficult ask. The underlying purpose of code is efficiency by reducing redundancy. Recalling Jeffrey, IDEs are conducive to contract 'reusability;' fostering an increase in 'base documents' and import of boilerplate clauses.[508] It may be unavoidable that clauses drafted in

[502] The firm comments on the uniqueness of the project from other 'code-fication' ventures. The project avoids mere replication and enforcement of existing legal agreements. *See* "DnA Contracts," Github King and Wood Mallesons (accessed May 2020), https://github.com/KingandWoodMallesonsAU/Project-DnA.

[503] *Id.*

[504] *Id.*

[505] *Id.*

[506] *Id.*

[507] Richard A. Posner, *The Law and Economics of Contract Interpretation*, 83 Texas L. Rev. 1581, 1587 (2005).

[508] Jeffrey, *supra* 424.

formal language become standard boilerplate; easily reusable in a number of agreements. Consequently, there remains the risk of a conceptual mismatch.

Drafting contracts both in natural language and code at inception is perhaps optimistic. To preserve the integrity and consistency of their contracts, KWM would be obligated to determine whether (a) the clause should be importable to other agreements; or (b) hybrid contracts should act as unique templates of their own. A workaround may be to create standards for contractual clauses conducive for 'code-ification;' as opposed to drafting in a combinatory manner. Regardless, the possibility of reframing contracts doctrine altogether is foreseeable.

B. The Code is Mightier than the Pen?

Though efficient, standardizing legal language has the potential of shifting the dynamics of contract negotiation and clause re-drafting. Consider Legalese's L4. The difficulty with the customized language – as one intended for legal writing – is that its default function has already translated legal language to code. It embodies specific assumptions of the law in its descriptive state. Parties using the L4 language then inherit such assumptions, changing their interpretation of contractual obligations and post-agreement behavior.

Perhaps Stanley Fish described it best, "language carries obligations and commitments that were once undertaken but eventually assumed; thereby rendering inseparable its original intentions at its core."[509] As a result, inherent philosophical and moral concepts are 'built into'[510] the language such that overtime its interpretative exercise is forgotten and accepted as fact. Similarly, Smith states, "...there are many [contractual] phrases requiring the assignment of an interpretation and the interpretations can interact in ways that are sometimes hard to foresee."[511] With the use of programming languages to draft contracts, the forthcoming challenges would be to ensure that the interpretative exercise is not forgotten; that meaning remains a continuum. Interpretation should allow for responsiveness to changing environments.

[509] Stanley Fish, *Is there a text in this class? The Authority of Interpretive Communities* 108 (1980).

[510] *Id* at 107.

[511] Smith, *supra* 419 at 1206.

Frank Pasquale reflects on interpretation by drawing on the "elective affinities between poets and lawyers."[512] He argues, "[t]he law is a rich source of metaphor for poetry"[513] that extends beyond technical expertise in its drafting. Pasquale warns of the "reductive demands of technology,"[514] whereby its competencies are limited to sets of commands and series of directives. Rather, the poetic construction of legal rules embodies a sensibility and sensitivity to circumstance that is necessary in legal writing.[515] As a result, the space for quantification and simplification of language stands in opposition to the inherent art of legal drafting. If polysemy is an integral feature of natural language that cannot be rid, how then could programming languages find its place in legal language?

C. Party Reactions

Understanding party behavior may be helpful. Zev Eigen reflects on contracts "in action."[516] Contracts hold the impression of legal constraints,[517] thereby specificity in language matters at the formation of the contract. In an empirical study, Eigen identifies two key propositions on questions of behavior around contracts. He states, contracts are a product of how drafters and signers interpret the law.[518] This reiterates the notion that *how* contracts are written frame the behavior of parties; drafting influences performance. As discussed, contract incompleteness signals trust between parties.[519] In contrast, standardized legal language is authoritative in character. It is the drafters' interpretation of the law; not the signers. In this case, programming languages risk eliminating the signers' altogether, and 'the drafters' are the code itself.[520]

---

[512] Frank Pasquale, *The Substance of Poetic Procedure: Law & Humanity in the Work of Lawrence Joseph*, 32 Law & Literature 1, 7 (2020). *See* also Pasquale's references to the similarities between lawyers and poets found in David Kader and Michael Stanford, *Poetry of the Law: From Chaucer to the Present* (2010).

[513] *Id.*

[514] *Id.* at 33.

[515] *Id.* at 34.

[516] Eigen, *supra* 476.

[517] *Id.* at 16.

[518] *Id.* at 7.

[519] Eigen et. al, *supra* 477

[520] Recall Lawrence Lessig and the conceptualization of code as law. Lessig draws attention to code as a form of control; that "code writers are increasingly lawmakers." *See* Lawrence Lessig, *Code 2.0* 79 (2006).

Online form-contracts[521] may be a revealing ancestor. In another study, Eigen tests the extent of party compliance with online form-contracts. The paper empirically examines contract enforcement on individuals relative to the framing of obligations and participation in drafting.[522] His findings note that the option to modify the terms and conditions positively impacts the eventual fulfilment of contractual obligations. To participate in the formation of the contract importantly distinguishes an individual's interpretation of contractual obligations. Participation transforms meaningless instructions to promises. Eigen states, "[p]romise creates obligation, whereas consent tolerates limits on what is being passively imposed or, [...] on rights surrendered."[523] The outcome of Eigen's experiment reveals that even the slightest effect into the contractual process is sufficient to demonstrate the heart of contracts doctrine: the will of contractual parties.

EMERGING FRONTIERS: NEXT STEPS

For programming languages to act as a legal language, party autonomy cannot be compromised. While the intention of program languages is not presumably to place limitations on contract formation, "law has language at its core."[524] Consequently, the functional nature of most programming languages has an inadvertently transformative impact on legal writing and the character of contract law. Next steps would require an untangling of performance from mutual assent.

a. New Encasings

For programming languages such as Solidity and Sophia, an easy fix may be to add legal effect to the annotations.[525] This would immediately reaffirm the weight of details in the contract; ensuring the role is understandably *prescriptive* as opposed to *descriptive*. Moreover, unintended transformations of contractual terms from conditions to warranties would be avoidable.

---

[521] Zev Eigen defines as online form-contracts as "contracts unilaterally drafted." *See* Zev J. Eigen, *When and Why Individuals Obey Contracts: Experimental Evidence of Consent, Compliance, Promise, and Performance*, 41 J. OF LEGAL STUDIES 67 (2012).

[522] *Id.* at 68.

[523] *Id.* at 90.

[524] Markou and Deakin, *supra* 410 at 3; and "...the central place of language in law" described in Pasquale, *supra* 512 at 31. *See* also Frank E. Cooper, *Effective Legal Writing* (1953) and his introduction with Law is Language.

[525] This, however, depends on the technical competency of lawyers to verify that annotations have been performed. Such an approach has been suggested by Shaanan Cohney and David Hoffman; the layering of the scripting and natural language to form a 'contract stack' whereby promises are 'legally-operative.' *See* Shaanan Cohney and David Hoffman, *Transactional Scripts in Contract Stacks*, U of Penn Inst for Law & Econ Research Paper No. 20-08, 40-60.

In the case of Blawx and Lexon, the question is more complex as rules, categories, and framing are intentionally reconfigured. Blawx and Lexon predicate on a shift in the performance of the law; bringing to light the translation of legal concepts. The adage, "the medium is the message," is particularly relevant for these languages. Both Blawx and Lexon express their own conceptual framework, redefining and asserting the meaning of existing legal interpretations. This further speaks to the limits of the law[526] and the difficulty with demarcating legal concepts.

Lessons of methodological transplant may be insightful. Katja Langenbucher engages with a theory of knowledge transfer that occurs between fields; subsequently, creating an import and inheritance between concepts. Langenbucher notes that the integration of economics, for example, in 'legalese' offers promises of (1) 'tested predictions;' (2) clear questions and precise methodology; and (3) a common language.[527] But, the difficulty of transplant, she suggests, is the superficiality of the import. This is typically owed to a misalignment between *assumptions* about the discipline and the method itself.

Similarly, programming languages such as Blawx and Lexon seek to offer comparable promises of clarity and precision. Their current state, however, could risk undercutting contracts doctrine as clauses are forcibly fit to what is permissible of the language as opposed to legal principles. For Blawx, the conflation of validity with enforceability is problematic. Lexon, on the other hand, constructs barriers for contracting parties by limiting the vocabulary and grammars available. Again, the language must be sufficiently agile to accommodate for the possibility of unpredictable circumstances. Ultimately, contracts are about regulating the future through transactions.[528] Contracts allow performance "to unfold over time without either party being at the mercy of the other [...]"[529] By confining the operational space, the 'medium' inadvertently ties the hands of its parties.

b. Recycled Structures

For languages like OpenLaw, the challenge is two-fold: (1) achieving balance between natural and symbolic (numeric) language; and (2) the simplification of legal writing. A hybrid language raises the

[526] As described by Joseph Raz as the exercise of distinguishing the principles and standards that should be included or excluded from the legal system. *See* Joseph Raz, *Legal Principles and the Limits of Law*, 81 Yale L. J. 823 (1972).

[527] Katja Langenbucher, *Economic Transplants: On Lawmaking for Corporations and Capital Markets* 8-9 (2017).

[528] Geoffrey Samuel, *The Reality of Contract in English Law*, 13 Tulsa L. J. 508, 523 (2013).

[529] Posner, *supra* 507 at 1582.

potential for parallel drafting. An initial assessment of clauses that may be 'code-ified,' thereby, become paramount to maintaining the integrity of the contract. This could foreseeably demand defining working guidelines on articles and provisions that are (1) invariant to context; and (2) for varying types of contracts. Smith's 'modular boilerplate' could be an excellent start; specifically, the combined assessment on the remoteness of the audience and risk of the transaction.[530]

Still, contracts must be "tailored to the parties' needs;"[531] and integrating standard 'reusable' code could occasionally lead to an improper fit. To have equal effect between natural language clauses and code, execution must mirror the qualitative description. Ron Dolin reflects on particular elements of contracts that are already "tagged, labeled, identified, or otherwise 'marked up'...[and] amenable to complex search and integration."[532] Existing tools, such as the Extensible Markup Language (XML), predefine rules for encoding documents to allow for both human and machine-readability. Even in cases where rules are not predefined, definition languages[533] outline permissible tags with attributes that are readily usable.

Dolin argues that the tradeoffs of using XML are largely between increased accuracy and reduced ambiguity against significant "upfront costs."[534] He suggests then that the difficulty of integrating XML in legal documents is unpacking the "intimate relationship between information needed to be exchanged [...] and the shared, controlled vocabulary used to express details."[535] He cites the example of medical informatics that thrived on XML integration. Their success, Dolin suggests, is owed to a standardized method of information exchange and "well-defined descriptions."[536] The question becomes: are there well-defined descriptions and a shared, controlled vocabulary in contract law?

Two examples are informative: (1) the OASIS LegalXML eContracts schema; and (2) the Y Combinator Series Term Sheet Template. OASIS, the Organization for the Advancement of Structured Information Standards, is a nonprofit consortium that works on the development of

---

[530] Smith, *supra* 419 at 1209 -1210.

[531] *Id* at 1210.

[532] Ron Dolin, "XML in Law: An Example of the Role of Standards in Legal Informatics," forthcoming paper, 2.

[533] *See* for example, Document Type Definition (DTD), XML Schema Definition (XSD) and Relax NG

[534] Dolin, *supra* 532 at 7.

[535] *Id.*

[536] *Id.* at 8.

standards across a wide technical agenda.[537] In 2007, a technical committee on contracts created an XML language to describe a generic structure for a wide range of contract documents. This became the OASIS LegalXML eContracts Schema (eContracts Schema). The intention of the eContracts Schema is to "facilitate the maintenance of precedent or template contract documents and contract terms by persons who wish to use them to create new contract documents with automated tools."[538] That is, the eContracts Schema focuses on reproducibility, reusability, and recursion.

Interestingly, the most striking feature of the eContracts Schema is their metadata component. Their model allows its users to add metadata at the contract and clause level for specific legal subject matter or categorization of distinct content. In this case, eContracts Schema provides an opportunity for clauses to cater to the specific requirements of contractual parties.

The Y Combinator Series A Term Sheet Template (Term Sheet)[539] is a standard form of terms to seek Series A funding.[540] The term sheet was drafted by Y Combinator, a venture investor that supplies earliest stage venture funding for startups.[541] The Term Sheet was created to inform founders of startups on terms most frequently negotiated, particularly when seeking funding for this next stage. The Term Sheet was drafted based on the experiences of venture investors. Not only does it provide a baseline for founders, but more importantly, it increases transparency about investors' perceived risks.[542]

Unlike the eContracts Schema, the Term Sheet is not 'technologically-driven.' Nevertheless, it illustrates that well-defined descriptions and a shared, controlled vocabulary exist in contracts. To a large extent, the Term Sheet is no different than any existing contract template. Yet, the most unique characteristic of the Term Sheet is the tone of the contract. Unlike other templates, the intention is

---

[537] "About Us," OASIS Open Standards. Open Source. (accessed August 2020), https://www.oasis-open.org/org.

[538] *See* Abstract section. "eContracts version 1.0," OASIS (accessed August 2020), http://docs.oasis-open.org/legalxml-econtracts/CS01/legalxml-econtracts-specification-1.0.html.

[539] *See* Appendix for Term Sheet.

[540] Series A funding is defined as funding to further refine the product and monetize the business, once a startup has established a user base with consistent performance. *See* Nathan Reiff, "Series A, B, C, Funding: How It Works," Investopedia (March 5, 2020), https://www.investopedia.com/articles/personal-finance/102015/series-b-c-funding-what-it-all-means-and-how-it-works.asp.

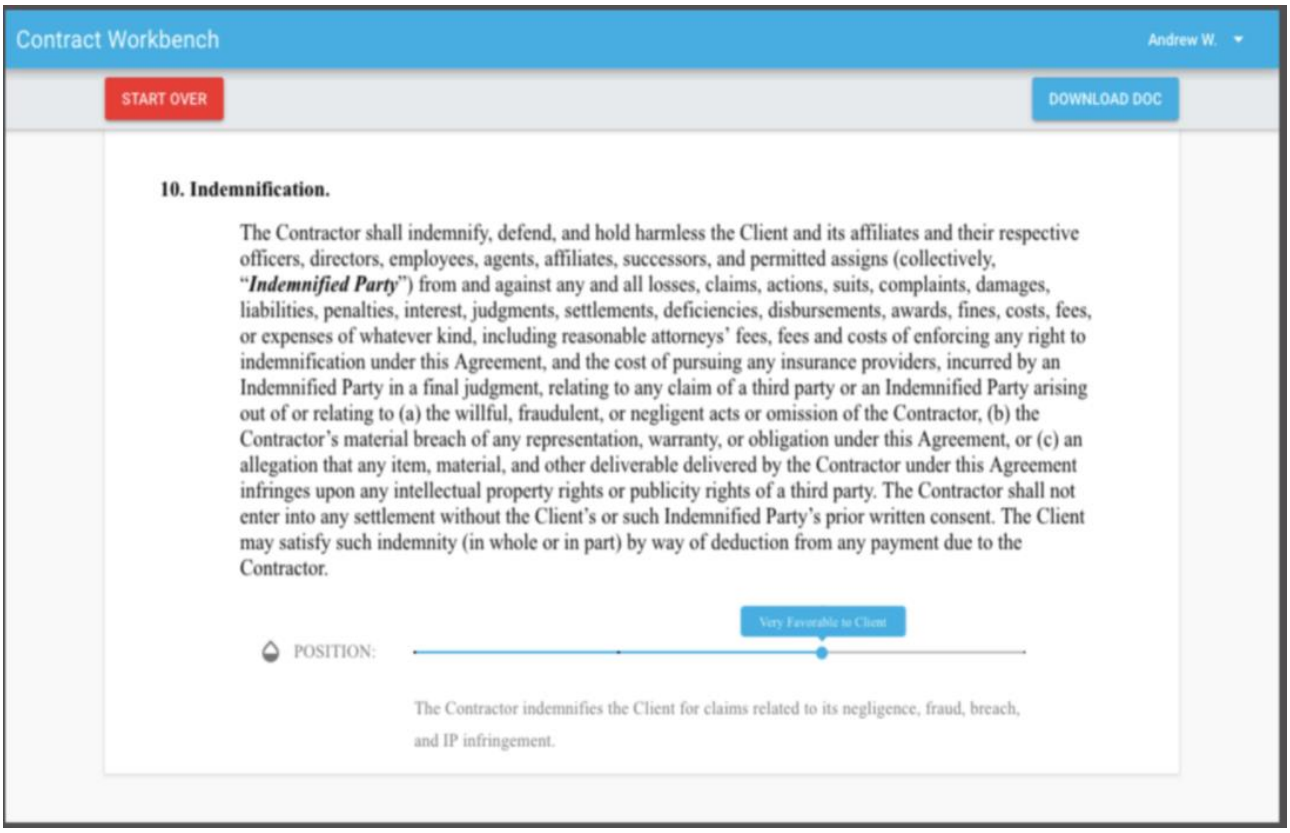
[541] "About Y Combinator," Y Combinator (accessed August 2020), https://www.ycombinator.com/about/

[542] "Series A Term Sheet Template," Y Combinator (accessed August 2020), https://www.ycombinator.com/series_a_term_sheet/

not to blindly assert 'boilerplate' contractual terms to drafters. Instead, the Term Sheet offers recommendations to support the positions of both contractual parties.

Recent Legal Tech startup, Lawgood, mirror the exact intentions of the Term Sheet: contract drafting based on verified expertise. Lawgood's drafting tool, the Contract Workbench, heightens the quality of the drafting process by developing a precedent language that tailors to the positions of the parties.[543]

Consider the sample indemnification clause drafted on Lawgood.



There are a number of fascinating features[544] to the software. Notably, Lawgood offers several drafting options depending on the needs of the contractual parties. The familiarity of MS Word is coupled with a toggle switch that highlights the most common positions negotiated when drafting indemnity clauses. Below the toggle, a 'simplified' version of the term explains the meaning of the various

[543] Lawgood (accessed August 2020), https://lawgood.io/product

[544] It should be noted that features of Lawgood extend beyond the toggle. There are also text buttons and embedded code. See *id.*

positions, distilling and translating the legalese to plain English. Unlike the examples of the programming languages studied in the paper, the translations are intended to be instructive rather than binding.

There are indubitably caveats to the software. The precedent language, created by Lawgood, draws primarily from the experiences of its developers. That is, it gathers the collective legal knowledge of contractual precedents specific to the expertise of its founders. The product is, therefore, limited to the frameworks as stipulated by its creators. Nonetheless, Lawgood illustrates that a marriage of the old and new is possible – in particular, the prospect of a shared lexicon in contract law.

All in all, hybrid programming languages, like OpenLaw, represent the recurring theme that there are distinct metaphorical spaces between determinacy and indeterminacy. Legal drafting is simplified through the act of sorting, assessing whether a clause is sufficiently amenable to reusability. From XML to Lawgood, the open secret is that contractual language will always remain a dialogic process between its parties.

To conclude, the purpose of the study is not to suggest that programming languages are not a possibility for legal writing. In fact, formal languages could provoke a more transparent discussion of obligations and expectations involved within the dynamics of contractual negotiation.[545] Yet, the mechanics of current programming languages illuminate that there is still work required for code to become a legal language. Geoffrey Samuel states, the "true meaning of a legal text is hidden within the language employed."[546] Reflecting on programming languages as a medium for contract drafting has revealed that language indeed could alter the function of contract law. Further discussion is required on *how* programming languages could better navigate and shape the legal landscape. For now, perhaps it can be understood that the tool is an extension of the craft, and not simply a means for its effectuation.

[545] Recall the discussion on modularity.

[546] Geoffrey Samuel, *Is Legal Reasoning like Medical Reasoning?*, 35 LEGAL STUDIES 323, 334 (2015).

3B- Object-Oriented Design of Legal Text (Judicial Decisions)

Rules are pervasive in the law. In the context of computer engineering, the translation of legal text to algorithmic form is seemingly direct. In large part, law may be a ripe field for expert systems and machine learning. For engineers, existing law appears formulaic and logically reducible to 'if, then' statements. The underlying assumption is that the legal language is both self-referential and universal. Moreover, description is considered distinct from interpretation; that in describing the law, the language is seen as quantitative and objectifiable. Nevertheless, is descriptive formal language purely dissociative? From the logic machine of the 1970s to the modern fervor for artificial intelligence (AI), governance by numbers is making a persuasive return. Could translation be possible?

Most recently, Douglas Hofstadter commented on the "Shallowness of Google Translate."[547] He referred largely to the Chinese Room Argument;[548] that machine translation, while comprehensive, lacked understanding. Perhaps he probed at a more important question: does translation *require* understanding? Hofstadter's experiments indeed seemed to prove it so. He argued that the purpose of language was not about the processing of texts. Instead, translation required imagining and remembering; "a lifetime of experience and [...] of using words in a meaningful way, to realize how devoid of content all the words thrown onto the screen by Google translate are."[549] Hofstadter describes the appearance of understanding language; that the software was merely 'bypassing or circumventing' the act.[550]

Yulia Frumer, a historian of science, notes that translation not only requires producing the adequate language of foreign ideas, but also the "situating of those ideas in a different conceptual world."[551] That is, with languages that belong to the same semantic field, the conceptual transfer in the translation process is assumed. However, with languages that do not share similar intellectual legacies, the meaning of words must be articulated through the conceptual world in which the language is seated.

---

[547] Douglas Hofstadter, *The Shallowness of Google Translate*, The Atlantic (January 30, 2018), https://www.theatlantic.com/technology/archive/2018/01/the-shallowness-of-google-translate/551570/.

[548] A thought-experiment first published by John Searle in 1980 arguing that syntactic rule-following is not equivalent to understanding.

[549] Hofstadter, *supra* 547.

[550] *Id.*

[551] Yulia Frumer, *Translating Worlds, Building Worlds: Meteorology in Japanese, Dutch, and Chinese*, 109 ISIS 326 (2018).

Frumer uses the example of 18th century Japanese translations of Dutch scientific texts. The process by which translation occurred involved first analogizing from Western to Chinese natural philosophy; effectively reconfiguring the foreign to local through experiential learning. This is particularly fascinating, provided that scientific knowledge inherits the reputation of universality. Yet, Frumer notes, "...if we attach meanings to statements by abstracting previous experience, we must acquire new experiences in order to make space for new interpretations."[552]

Mireille Hildebrandt teases this premise by addressing the inherent challenge of translation in the computer 'code-ification' process. Pairing speech-act theory with the mathematical theory of information, she investigates the performativity of the law when applied to computing systems. In her analytical synthesis of these theories, she dwells on meaning. "Meaning," she states, "...depends on the curious entanglement of self-reflection, rational discourse and emotional awareness that hinges on the opacity of our dynamic and large inaccessible unconscious. Data, code...do not attribute meaning."[553] The inability of computing systems to process meaning raises challenges for legal practitioners and scholars. Hildebrandt suggests that the shift to computation necessitates a shift from reason to statistics. Learning to "speak the language" of statistics and machine learning algorithms would become important in the reasoning and understanding of biases inherent in legal technologies.[554]

More importantly, the migration from descriptive natural language to numerical representation runs the risk of slippage as ideas are (literally) 'lost in translation.' Legal concepts must necessarily be reconceptualized for meaning to exist in the mathematical sense. The closest in semantic ancestry would be legal formalism. Legal formalists thrive on interpreting law as rationally determinate. Judgments are deduced from logical premises; meaning is assigned. While, arguably, the formalization of law occurs 'naturally' – as cases with like factual circumstances often form rules, principles, and axioms for treatment – the act of conceptualizing the law as binary and static is puzzling. Could the law behave like mathematics; and thereby the rule of law be understood as numeric?

---

[552] *Id.* at 327.

[553] Mireille Hildebrandt, *Law* as *computation in the era of artificial intelligence: Speaking law to the power of statistics*, Draft for SPECIAL ISSUE U. TORONTO L.J. 10 (2019).

[554] Advances in natural language processing (NLP), for example, have opened the possibility of 'performing' calculations on words. This technology has been increasingly applied in the legal realm. See *id.* at 13.

Technology not only requires rules to be defined from the start, but that such rules are derived from specified outcomes. Currently, even with rules that define end-states, particularized judgments remain accessible. Machines, on the other hand, are built on logic and fixed such that the execution of tasks becomes automatic. Outcomes are characterized by their reproductive accuracy. Judgments, on the other hand, are rarely defined by accuracy. Instead, they are weighed against social consensus.

To translate the rule of law in a mathematical sense would require a reconfiguration of legal concepts. Interestingly, the use of statistics and so-called 'mathematisation' of law is not novel. Oliver Wendell Holmes Jr. most famously stated in the *Path of Law* that "[f]or the rational study of the law, the blackletter man may be the man of the present, but the man of the future is the man of statistics and the master of economics."[555] Governance by numbers then realizes the desire for determinacy; the optimization of law to its final state of stability, predictability, and accuracy. The use of formal logic for governance has a rich ancestry. The common denominator was that mathematical precision should be applied across all disciplines.

Legal texts, then, may arguably be represented as computational data with terms made 'machine-readable' through a process of conversion. Despite the capacity to express legal language in an alternative computable form, the notion of interpretation appears to have changed. How would digital data inscription and processing alter methods of legal reasoning?

a. Outline of Approach

The case study follows a fundamentally semantic conundrum: what is the significance of 'meaning' in legal language? From a statistics standpoint, meaning can be approximated. Applying word analogies as the 'mathematical' basis, meaning is gauged by the statistical probability of the response. In recognizing the context and relationship between words, meaning hinges on the frequency of its appearance in a particular setup. That is, what do its neighbors reveal about the word in question?

Reflecting on Hildebrandt and Frumer, meaning is associated with experience; thereby finding meaning to legal concepts would require abstracting from experience. Should experience be built from conceptual worlds, to move across these worlds would be to translate. Translating legal language

[555] Oliver Wendell Holmes Jr., *The Path of Law*, 10 HARV. L. REV. 457, 469 (1897).

then requires a reframing of legal concepts; perhaps an expression of the law based on statistical experience as opposed to natural language.

The project will proceed in two phases: (1) the proof of concept (POC);[556] and (2) application to broader legal corpora. In the first phase, the POC will analyze three United States Supreme Court cases. The selection was chosen on the basis of a similar factual premise and time frame. That is, all three cases involve defining the use of firearms and were ruled in rapid succession. These cases are *Smith v. United States* (1993), *Bailey v. United States* (1995), and *Muscarello v. United States* (1998). While there are evidently a number of caveats[557] to this selection, it nonetheless has merit as an interesting starting point. Notably, the POC wrestles with the existence of legal concepts. The goals of the POC are two-fold: (1) to analyze the processes involved with legal interpretation and reasoning; and (2) critically assess them against the function of law.

Methodologically, the POC tests translation by deconstructing sentences from existing legal judgments to their constituent factors. Definitions are then extracted in accordance with the interpretations of the judges. The intent is to build an expert system predicated on alleged rules of legal reasoning. I intend to apply both linguistic modelling and natural language processing (NLP) technology to parse the legal judgments. The preliminary hypothesis is that, by analyzing the components of legal language with a variety of techniques, we can begin to translate law to numerical form. Furthermore, it would be interesting to consider what contextual understanding may need to exist to understand the language of various legal documents.

Following the POC, I will extend the test to a larger corpora of case law. This stage of the research will consider the feasibility of expanding the approach to similar legal texts. For the purposes of the current case study, I focus on the observations and findings from the POC. Though microscopic in the landscape of United States jurisprudence, initial observations appear to suffice in contributing to a more fruitful dialogue on the integration of computational technology in law.

The POC falls in line with existing literature on Law2Vec and legal word embeddings. Equally, the project extends beyond prior research in the area, combining a broadly statistical model of context

[556] As mentioned, the second case study is seated with an ongoing interdisciplinary project. Therefore, the second case study and my observations are, in fact, largely drawn from the POC.

[557] Some of these caveats include selection bias, sample size, and perhaps more importantly, an amendment has since been made to the legislation in question.

with the relative precision of syntactic structure. In effect, the POC intends to generate building blocks to determine "context" explained in the text; thereby, able to define the use of firearms through a framework of extraction.

The case study will proceed as follows. Part I will begin with a literature review of texts that have fueled the project's inquiries and formed the environment which it intends to resolve. As the nature of the cases study is fundamentally interdisciplinary, it draws reference from law, linguistics, and computer science. Part II discusses the methodology we have taken; highlighting both elements of inspiration and strategies considered. Part III teases at preliminary observations and notes of interest during the project's progression. Part IV details the technological implementation and the actual steps towards translation. Part V reflects on early achievements and areas of further advancement. I will then conclude with a few final remarks.

It must also be noted that, throughout the case study, I frequently move between the use of "I" and "we." This is because the case study relies on methods that were a result of the broader interdisciplinary collaboration. I stress that, without the insight and contribution of the data scientist, mathematician, and linguist in our project team, the perspectives and observations from this case study would not have been possible.

I. LITERATURE REVIEW

a. Jurisprudential Heritage

AI adjudication is an evidently polarized subject. Questions around the prospect of "robot judges" typically center on morality and equitable justice;[558] on issues of explanability and Black Box machine learning.[559] In common law systems, the art of drafting legal opinions begins with mastering legal argumentation. To ground the argument within the sphere of existing legal texts is the linchpin of judicial decisions.

[558] Richard M. Re and Alicia Solow-Niederman, *Developing Artificially Intelligent Justice*, 22 STAN. TECH. L. REV. 242 (2019).

[559] *See* Yavar Bathaee, *The Artificial intelligence Black Box and the Failure of Intent and Causation*, 31 HARV. J OF L. & TECH 890; and also, Frank Pasquale, *Black Box Society: The Secret Algorithms that Control Money and Information* (2015).

Legal theory becomes a referencing point when courts are asked to interpret legal documents. Textualism, for example, "narrow[s] the range of acceptable judicial decision-making and acceptable argumentation"[560] by turning to dictionary definitions and rejecting judicial speculation. Yet, what is the purpose of 'narrowing the range'? To that question, Antonin Scalia answers, "...textualism will provide greater certainty in the law, and hence greater predictability..."[561] So, what are its assumptions and implications? Eric Posner suggests, there may be aspirational intentions "to keep the law pure";[562] or otherwise, to ensure that the legal system is consistent. Textualism also reinforces the role of judges. That is, judges are to interpret passively, and that legal interpretations are to be semantic.[563]

Consider the infamous example of a municipal legislation stating that "no person may bring a vehicle into the park."[564] Would an ambulance be permitted to enter the park in the event of an accident? For textualists, they may argue that – according to the dictionary definition – an ambulance is a vehicle; and thereby, cannot enter the park. Should the legislators have thought an ambulance was an exception, they would have included it in the text. Accepting the premise of that argument, what about a police car or a firetruck? Perhaps the legislation should be amended to include all emergency vehicles. What happens then if an ambulance is merely parked inside the park with no foreseeable emergency?

The example illustrates that the problem with textualism becomes rapidly cyclical, as interpretations rendered must either become increasingly narrow or increasingly broad to accommodate a "myriad [of] hypothetical scenarios and provide for all of them explicitly."[565] Textualism, therefore, falls down the slippery slope of literalism. Words of legal texts are assumed to embody intrinsic meaning and are waiting to be extracted.

---

[560] Antonin Scalia and Bryan A. Garner, *Reading Law: The Interpretation of Legal Texts* xxvii-xxix (2012).

[561] *Id.*

[562] Eric Posner and Adrien Vermeule*, Inside or Outside the System?*, 80 U. CHI. L. REV. 1743, 1775 (2013).

[563] Richard A. Posner, *The Incoherence of Antonin Scalia*, New Republic (August 24, 2012), http://www.newrepublic.com/node/106441/print.

[564] Taken originally from H.L.A. Hart where he posed the hypothetical of "no vehicles in the park." *See* H.L.A. Hart, *Positivism and the Separation of Law and Morals*, 71 HARV. L. REV. 593, 607 (1958). This has often been referenced in legal literature. *See* for example, Pierre Schlag, *No vehicles in the Park*, 23 SEATTLE U. L. REV. 381, 382 (1999); and more recently, Michael Genesereth, *Computational Law: The Cop in the Backseat*, White Paper, CodeX: The Center for Legal Informatics (2015), available at: http://logic.stanford.edu/publications/genesereth/complaw.pdf.

[565] Posner, *supra* 563.

Moreover, the impact of mere 'extraction' is its precedential value. The approach, taken most prominently in common law systems, is to follow past decisions. Adopting the decisions of the past to guide future conduct parallels this exact act of extraction. That is, applying past precedents provides the scope for a "gradual moulding of the rules to meet fresh situations as they arise."[566] Decisions have binding legal force. Interpretations of the past should carry the definitions to be used moving forward. The role of the judge is that of an archaeologist; excavating legal truths from judicial past.

This is seemingly straightforward. Yet, the challenge encountered is identifying within the decision the kernel of precedent. Holmes describes the challenge as a paradox of form and substance in the development of the law. The form is logical, as "each new decision follows syllogistically from existing precedents."[567] Still, its substance is legislative and draws on views of public policy. Holmes argues that the law is driven by the "unconscious result of instinctive preferences and inarticulate convictions;" and therefore, "the law [is] always approaching, and never reaching, consistency."[568] The ostracized conclusion would be that judicial decisions have an element of inexplicability, and are, in fact, a 'Black Box.'[569] Recalling Hildebrandt, "meaning" becomes a metaphor and the heart of the juridical process.

The significance of the paper is, in part, to unpack the paradox articulated by Holmes. The selected cases aim to paint a picture on the use of precedent as a legal tool; and whether the law subconsciously follows a logic. To create the painting, I again draw inspiration from the field of linguistics.

b. Linguistic Influence

A grasp on the underlying hierarchical structure of language is key to breaking down sentences in a meaningful manner. To recall, analyses of sentence structure fall primarily into two schools of

---

[566] *See* chapter on Theories of Adjudication, in particular the discussion on *stare decisis* as the 'life blood of legal systems,' requiring precision in addition to stability and certainty. Michael Freeman, *Lloyd's Introduction to Jurisprudence* (9th ed., 2014).

[567] Oliver Wendell Holmes, *The Common Law* Lecture I: Early Forms of Liability (Project Gutenberg eBook, 2000), available at: https://www.gutenberg.org/files/2449/2449-h/2449-h.htm#link2H_4_0001.

[568] *Id.*

[569] *See* for example, Dan Simon, *A Third View of the Black Box: Cognitive Coherence in Legal Decision Making*, 71 U. CHI. L. REV. 511 (2004).

thought: (1) dependency; and (2) phrase structure. The former, commonly represented as dependency trees, begins with the root verb of the superordinate clause and branches out from there, with subordinate verbs arranging substructures. Dependency trees map one node to each word without projecting constituent phrases: each word simply depends on another. For example, in most English sentences, the subject typically falls to the left of the verb, while its other dependencies (e.g. its objects) fall to the right. Since each word in a dependency syntax is represented by precisely one node, structural redundancy is arguably decreased. This system has been characterized as well-suited for algorithmic translation from natural language, owing to the node conservatism and predictability of anchoring sentences through its verbs.

Alternatively, phrase-structure representations, notably spearheaded by Noam Chomsky,[570] use constituency relations. In contrast with dependency trees, each 'constituent' (or, individual element) in a sentence is headed by its own phrasal node. Subsequently, purely binary branching can occur. The elegance of these representations is that they work generatively. That is, even a small selection of rules can produce a wide variety of structures found across natural language. Furthermore, constituency embraces analysis of underlying structure and transformations, accounting for numerous phenomena such as subject-verb inversion in interrogatives.[571] Phrase structure also permits a powerful structured analysis of syntactic relationships.[572]

Semantic form traditionally involves the classical theory of concepts, otherwise known as definitionism or componential analysis. Here, semantic meaning is encapsulated as a combinatorial set of true/false statements, akin to a checklist of conditions. For example, *apple* might be composed of *+fruit, +green, +round*. Classical theory, therefore, considers the componential elements from which semantic meaning is formed, allowing for a systematic view on word-to-word relationships and validity.[573]

---

[570] Noam Chomsky, "Remarks on Nominalization," in R.A. Jacobs and P.S. Rosenbaum (eds.), *Readings in English Transformational Grammar* 184-221 (1970).

[571] Subject-verb inversion is the phenomenon whereby the verb is raised to a position in front of its subject, signalling an interrogative: "Have you seen my dog?". This raising is seen as a transformation.

[572] For example, the c-command relationship is easily identified, which is particularly useful when managing anaphora resolution through Government and Binding Theory (GBD). *See* Andrew Carnie, *Syntax: A Generative Introduction* (3rd ed. 2012).; and also Ray Jackendoff, *X Syntax: A Study of Phrase Structure* (1977).

[573] For further details: the classical theory of concepts presents a deconstructive view of meaning (semantics). By breaking words down into sets of necessary and sufficient conditions from a set of meta-concepts, we view their 'true' definition and form comparisons. For example, *bachelor* and *husband* suggest a commonality of *+male* but a

However, classical theory is often criticized for its failure to account for phenomena such as the subjectivity or typicality of definitions.[574] Ludwig Wittgenstein posited, through his analogy with 'family resemblance,' an underlying prototype theory of concepts; as opposed to a fixed set of composite definitions. The claim is that some concepts are regarded more 'typical' of a category than others. For example, a robin is a more prototypical bird than an emu or a penguin. Consequently, these observations must be factored into the linguistic system.[575]

What further complicates the matter is the incongruence between semantics and pragmatics: the former concerns language independent of real-world context, whereas the latter is hinged upon situational context. Essentially, pragmatics is the application of semantics within context.[576] Consider the phrase, "it's rather chilly in here." Semantically, the meaning of the phrase is perhaps that, according to the speaker, "there is a place X in which the temperature is lower than is comfortable." Given the knowledge that the phrase was taken from a dialogue between two individuals, the phrase pragmatically could mean "please close the window for me;"[577] the reason for the choice of phrasing is likely owed to courtesy. More importantly, this form of expression is indicative of the flexibility of language and its inseparability from context: context contributes to meaning.

While semantics concerns the inherent and invariant properties of words and their combinations, pragmatics progresses into the realm of context and implicatures. Consequently, pragmatics in the context of NLP is seen as problematic: expert systems do not have the ability to infer extended meaning from context. Interestingly, legal texts are often regarded as rather structural, and perhaps even devoid of pragmatic content. Given the aforementioned premise, is legal language anchored exclusively in semantics? If so, how amenable is legal language to NLP analysis?

c. Technological Staging: AI and Law

distinction in the condition of *±married* (*-married* in the former and *+married* in the latter). *See* Eric Margolis and Stephen Laurence, *The Blackwell Guide to Philosophy of Mind* Concepts 190-213 (2003).

[574] *See* Ludwig Wittgenstein, *Philosophical Investigations* (2nd ed. 1953).

[575] Eleanor Rosch and Carolyn B. Mervis, *Family resemblances: Studies in the internal structure of categories*, 7 COGNITIVE PSYCHOLOGY 573 (1975).

[576] Keith Allan and Kasia M. Jaszczolt (eds.), *The Cambridge Handbook of Pragmatics* (2012). *See* also, Kasia Jaszczolt, *Semantics and pragmatics: Meaning in language and discourse* (2002).

[577] This is largely in line with discussions on conversational implicatures. *See* for example Henry E. Smith, *Modularity in Contracts: Boilerplate and Information Flow*, 104 MICH. L. REV. 1175, 1205 (2006).

Evidently, inspiration from linguistics is not novel as "law has language at its core."[578] As mentioned in the first case study, Markou and Deakin point to NLP as a powerful driver towards the emergence of Legal Tech. They identify the pressure points at which computability falls short; and where the legal system is incompatible with computer science.

To recall, they cleverly evoke Chomskyan and rationalist approaches to designing "hard-coded rules for capturing human knowledge."[579] Chomsky's work stirred further developments in NLP, eventually powering advances in machine translation and speech recognition. These advances, undoubtedly, were enabled by Deep Learning[580] models that were able to abstract and build representations of human language. Albeit the significant leaps brought on by such technologies, the threat discussed by Markou and Deakin stems from an underlying anxiety against "the epistemic and practical viability of using AI and Big Data to replicate core aspects and processes of the legal system."[581]

Subsequently, their reimagining of a legal system – one predicated on a hyper-formalized method of reasoning[582] – warns of the conceivable incongruence with the current normative legal structure. Using employment status as a test case, their paper explores first similarities between legal processes and machine learning technology. They note two key parallels: (1) abstraction to conceptual categories; and (2) error correction and dynamic adjustment.

Nevertheless, their thesis, or claim of divergent paths, is the quality of reflexivity[583] in legal knowledge. That is, legal categories both shape and are shaped by the "social forms to which they relate."[584] In

[578] Christopher Markou and Simon Deakin, *Ex Machina Lex: The Limits of Legal Computability*, Working Paper (2019), available at SSRN: https://ssrn.com/abstract=3407856. *See* also Frank E. Cooper, *Effective Legal Writing* (1953) and his introduction with Law is Language; and "...the central place of language in law" described in Frank Pasquale, *The Substance of Poetic Procedure: Law & Humanity in the Work of Lawrence Joseph*, 32 LAW & LITERATURE 1, 31 (2020).

[579] *Id. See* also cited reference, E Brill and RJ Mooney, *Empirical Natural Language Processing*, 18 AI MAGAZINE 4 (1997).

[580] Deep Learning is a subset of machine learning that involves artificial neural networks and the assigning of numerical weights on input variables. *See* a further explanation in Markou and Deakin, *supra* 578 at 10-12.

[581] *Id.* at 16.

[582] *Id.*

[583] Markou and Deakin reference Geoffrey Samuel's discussion of the perception, construction and deconstruction of fact. See *id.* at 29. *See* also Geoffrey Samuel, *Epistemology and Method in Law* (2003).

[584] *Id.*

other words, the existence of such categories is dependent on the force of law;[585] that there is continual reference between the law and its socially complex environment. The law cannot be divorced from its societal embedding. As a result, the law could never be descriptive, but rather 'naturally' prescriptive. Markou and Deakin, therefore, identify a fundamental philosophical mismatch as opposed to a structural, process-oriented incongruity. Their conclusions underline legal reasoning as beyond the straightforward application of rules to facts. Adjudication is a means of "resolving political issues."[586] For Markou and Deakin, there is no exact science to judicial decisions "because of the unavoidable incompleteness of rules in the face of social complexity."[587] Judgments could only 'approximate' from historical precedent. Translation of legal categories into mathematical function is, thus, not possible since the flexibility and contestability of natural language cannot be completely captured by algorithm.[588]

Holmes's paradox resurfaces. Holmes notes, to "attempt to deduce the corpus from *a priori* postulates, or fall into the humbler error of supposing the science of the law reside[s] in the *elegantia juris*, or logical cohesion of part with part"[589] mistakenly interprets law as systemically formalistic. While the issues identified by Markou and Deakin are undeniably significant, their arguments rely on the premise of a systems replication. That is, they warn of the project to replace entirely juridical reasoning with machine learning. Accordingly, there are sweeping inferences on the incompatibility of AI and law, bringing to light only one side of Holmes's paradox: the law is syllogistic in form.

Yet, there may be merit to an analysis at a micro-level. Programming languages may be able to perform the demands called upon for the functioning of society. Acknowledging that language is both constitutive of law and capable of realizing foundational rule of law principles, we again reassess the translation of natural language to computer code. The law hinges on complicated social and political relationships;[590] and more importantly, metaphors that require latent understanding of

[585] The 'force of law' refers to HLA Hart's argument that the power of legal institutions and the laws created by such institutions exist in virtue of a rule of recognition implicit in the practice of judges. *See* Gerald J. Postema, *Implicit Law*, 13 LAW AND PHILOSOPHY 361 (1994).

[586] Markou and Deakin, *supra* 578 at 30.

[587] *Id.* For further discussion on the 'incompleteness' and indeterminacy of the law, *see* Katherina Pistor and Chenggang Xu, *Incomplete* Law, 35 NYU J. INT'L L. & POL. 931 (2003).

[588] *Id.* at 33.

[589] Holmes, *supra* 567.

[590] Frank Pasquale, *A Rule of Persons, Not Machines: The Limits of Legal Automation*, 87 GEO. WASH. L. REV. 2, 6 (2019).

temporal societal constructs.[591] This suggests there may be a space to regard AI as complementary,[592] rather than substitutive, of legal actors. The key is to employ the proper language game.[593]

To return to Markou and Deakin, their arguments repeatedly point to the model of 'legal singularity.'[594] 'Legal singularity' draws from an association of the law as precise, predictable, and certain in its function.[595] The complexity of developments in machine learning for law suggests that legal singularity could be achievable.

In a vibrant thought experiment, Casey and Niblett suggest that existing legal forms will become irrelevant as machines enable the development of a new type of law: the micro-directive. The micro-directive is conceptually a new linguistic form, offering "clear instruction to a citizen on how to comply with the law."[596] In this futuristic construct, lawmakers would only be required to set general policy objectives. Machines would bear the responsibility to examine its application in all possible contexts, creating a depository of legal rules that best achieve such an objective. The legal rules generated would then be converted into micro-directives that subsequently regulate how actors should comply with the law.

Imagining the legal order as a system of micro-directives, the law finds itself drawn to a linguistic structuralist framework; carrying forth the jurisprudential work of Kelsen and the "pure science of law."[597] Just as a norm expresses not what is, but what *ought* to be – given certain conditions – the micro-directive draws attention to the semiotics of legal argument. Like Kelsen's norms, the micro-directive rests on the principle of effectiveness. The legal order relies on the assumption of being efficacious, such that its citizens conduct themselves in pure conformity with it.[598] But, on what

---

[591] Neil M. Richards and William D. Smart, "How should the law think about robots?" in Ryan Calo et al, eds, *Robot Law* 16-18 (2016).

[592] *See* for example Pasquale, *supra* 590.

[593] This is interpreted under the framework put forward by Wittgenstein. Wittgenstein regarded language as a form of life, and linguistic expression as constructive of its being. Conceivably, language could be no more than a list of orders and classifications. In abiding by the rules of association—or, to play the game—is to accept the inherent authority of its practice. *See* Wittgenstein, *supra* 574 at 11.

[594] Benjamin Alarie, *The Path of the Law: Towards Legal Singularity*, 66 U. TORONTO L.J. 443, 445 (2016).

[595] *See* Freeman, *supra* 566.

[596] Anthony J. Casey & Anthony Niblett, *The Death of Rules and Standards*, Coase-Sandor Working Paper Series in Law and Economics No. 738 (2015).

[597] *See* for general theory on 'science' of law, Hans Kelsen, *Pure Theory of Law* (1967).

[598] Hans Kelsen, *What is Justice?* 268 (1957).

principle? The micro-directive rests on a 'law and economics'[599] framework of effectiveness. Seated within the technical authority of AI,[600] the micro-directive distorts the realities of legal reasoning by removing value judgments from the adjudication process. The presumption that machines are able to generate neutral sets of information, then translate such information into perfectly comprehensible instruction, is evidently misinformed. It stands on the premise that translation operates without interpretation. More importantly, it strategically excludes the actors involved in the translation; inadvertently, conferring the rule of law to code. The process of transforming a general standard to a micro-directive is, therefore, a process of subverting politics in its linguistic casing.

So, how then could code become the vehicle that shapes the law? In practice, the most obvious example is traffic laws and speed regulation. Traffic lights "communicate the content of a law to drivers at little cost and with great effect."[601] The traffic light is regarded as translating legal complexity to a simple command. Traffic lights are increasingly being equipped with algorithmic technology to reflect real-time traffic flow and, accordingly, adjust the timing of light changes.[602] Moreover, traffic lights may soon include sensors that could appropriately identify patterns of distress and types of vehicles to allow for expedited changes in the event of emergency.

For Casey and Niblett, predictive models provide the content of the law. Micro-directives would then communicate the legal treatment of the particular conundrum.[603] Legal actors would equally rely on such models to assess the acceptable plans of action for a particular diagnosis or factual circumstance. The micro-directive then reinvents the legal system, as legal language is eradicated and bears a different linguistic form.

Though at polar ends of the spectrum, both Markou and Deakin and Casey and Niblett depend on the same underlying assumption of a wholesale replacement of legal reasoning. This approach certainly raises significant metaphorical eyebrows on the broad impacts of AI in law. It, however,

---

[599] Consider the argument for efficiency in common law rules (i.e. emergence of the economic loss 'rule') in Anthony Niblett et. al, *The Evolution of a Legal Rule*, 39 J. OF LEGAL STUDIES 325, 328-331 (2010).

[600] *See* discussion on algorithms as providing a convenient source of authority; trusting tasks to be controlled by technology and the delegation of responsibility. *See* Hannah Fry, *Hello World: Being Human in the Age of Algorithms* 16 (2018).

[601] Casey and Niblett, *supra* 596 at 18. *See* also, Sheila Jasanoff, *The Ethics of Invention: Technology and the Human Future* (2016).

[602] *Id.* at 19.

[603] Casey and Niblett, *supra* 596 at 16.

also avoids the nuances of the law that demand further analysis, in particular, the act of translation. Holmes described the "single germ multiplying and branching into products as different from each other as the flower from the root."[604] Thus, to make sense of the consequences of computational technology in law necessitates not an evaluation of the flower or the root, but the single germ.

Precision has often been argued as an essential component of legal language. Nonetheless, new factual circumstances create room for interpretation. How then could code-ification occur to account for an ever-adaptive, and evolutionary, system? In the following section, I will outline the computational tools used in the translation process. More importantly, I peel back the curtain behind translation, specifically, the decisions taken in the parsing of the legal judgments.

II. METHODOLOGY

Prior literature on Deep Learning in legal text analytics traditionally discussed crafting knowledge bases to capture legal concepts and terminology.[605] Ilias Chalkidis and Dimitrios Kampas reflect on existing techniques, but push the envelope by building word embeddings[606] trained over a large body of legal documents; a corpora composed of legislation from the UK, EU, Canada, Australia, USA, and others.[607] Applying the Word2Vec model,[608] Chalkidis and Kampas's own model – aptly named Law2Vec – offer a pre-trained set of legal word embeddings. Broadly, the process involves translating legal text to numeric form in order to calculate the relationships between legal terms. The calculation represents the probabilistic likelihood of one term appearing synonymous in the presence of the other. The main assumption is that "similar words tend to co-occur in similar contexts."[609]

---

[604] Holmes, *supra* 567.

[605] *See* for example Phong-Khac Do et al., *Legal Question Answering using Ranking SVM and Deep Convolutional Neural Network*, TENTH INTERNATIONAL WORKSHOP ON JURIS-INFORMATICS (2017), available at: https://arxiv.org/abs/1703.05320.

[606] Defined as a numeric representation of words whereby words with similar meanings would have similar representations. *See* Jason Brownlee, *What are Word Embeddings for Text?*, Deep Learning for Natural Language Processing (October 11, 2017), available at: https://machinelearningmastery.com/what-are-word-embeddings/.

[607] Ilias Chalkidis and Dimitrios Kampas, *Deep Learning in law: early adaptation and legal word embeddings trained on large corpora*, 27 ARTIFICIAL INTELLIGENCE AND LAW 171 (2018).

[608] A statistical model first introduced in 2013 that described two different algorithms used to process text into vectors. It is an example of the transformation of words to numeric form. *See* Mikolov et al., *Distributed representations of words and phrases and their compositionality*, PROCEEDINGS OF THE 26TH INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS (2013).

[609] Chalkidis and Kampas, *supra* 607 at 173.

Below is a table of a selected 20 words and their associated terms identified by the model:

article	convention, section, articles, clause, provisions
act	statute, provision, mccarranferguson, irca, tvpa
action	suit, actions, lawsuit, claim, proceeding
crime	offense, murder, crimes, felony, violent
felony	offense, misdemeanor, felonies, offenses, convicted
punishment	penalty, punishments, sentencing, sentence, imprisonment
security	social, health, administration, retirement
fraud	fraudulent, theft, deceit, misrepresentation, bribery
privacy	confidentiality, communications, liberty, freedom, freedoms
intellectual	copyrights, patents, copyright, trademark, wipo
terrorism	terrorist, trafficking, counter-terrorism, violent, laundering
immigrant	immigrants, nonquota, alien, asylum, citizenship
illegal	unlawful, corrupt, improper, illicit, fraudulent
drugs	drug, narcotic, addicts, psychotropic, medicines
appeal	appeals, review, hearing, appellate, appealed
abuse	violence, sexual, self-destructive, assault, mistreatment
alcohol	liquor, spirits, intoxicating, beer, vinous
complaint	grievance, allegations, allegation, complaints, counterclaim
indictment	conviction, summary, imprisonment, indictable, triable
motion	motions, petition, dismiss, leave, cross-motion

Table 1 Sample Legal Word Embeddings (Chalkidis and Kampas, *supra* 607 at 176)

This is undoubtedly remarkable. The associations made between the identified legal terms are indicative of the competence of machine learning algorithms for the analysis of complicated legal texts. Most fascinating perhaps are the terms associated with the word 'immigrant' found by the algorithm. Beyond locating synonyms, the terms deemed as similar reveal the latent politics of labelling that have classified immigrants as akin to aliens. Nevertheless, Chalkidis and Kampas offer only a limited perspective on legal concepts. The terms marked as 'legal' provide a scope of the law that does not consider the inherent interpretative exercise performed in adjudication. The act of legal reasoning is not represented. While Chalkidis and Kampas tease at the possibility of translation, the issue rather is arriving at the association. Chalkidis and Kampas could only bring to light the calculated similarities between legal terms; but they do not unpack *how* the similarity came about. In other words, the underlying process of deriving meaning is never exposed.

Moreover, the selection of terms deemed 'legal' are rather shallow. They are suggestive of a legal vocabulary, but do not probe at the function of these words. Taking again inspiration from literature outside of the legal realm, I focus on the mechanics of linguistic reasoning and the adjudicative process.

a. Technical Inspiration

As an introductory note on method, Markou and Deakin have helpfully outlined NLP technologies that have set the sail on current applications of AI-based innovations.[610] NLP is a combined scientific and engineering exercise, applying "cognitive dimensions of...natural language" to "practical applications...[of] interactions between computer and human languages."[611] For the intentions of the paper, the focus will be on natural language in written form; otherwise, text. Not only is it the form in which law most typically resides, text is also the observable component of language that exists in symbolic form.[612] Interestingly, mathematics– or to recall, the mental alphabets of Leibniz and Boole – is described as *the* symbolic language.[613] It follows that translation is most feasibly comparable where both 'languages' are in a similar state.

In order for natural language text to be 'primed' for translation, we applied an approach first introduced in the sphere of bioinformatics. In 2006, Fundel et. al. developed RelEx, or the relation extraction of free text, to better understand the interactions between genes and proteins marked by existing biomedical publications. RelEx relies on natural language *preprocessing*, "producing dependency parse trees and applying a small number of simple rules to these trees."[614] RelEx extracts qualified relations from natural language text by first breaking down sentences into component words (tokens), then uses a parser[615] to create syntactic dependency trees. These dependency trees are then leveraged from group tokens into 'noun-phrase' chunks.[616] Qualified relations are observed based on rules applied to dependency trees and their original sentences; which are then subjected to 'filtering.'[617] These rules would draw paths that connect known proteins that interact with one another.

---

[610] Markou and Deakin, *supra* 578 at 11.

[611] *Id.*

[612] *Id.* at 12.

[613] *See* literature: Ladislav Rieger, *Algebraic Methods in Mathematical Logic* The Language of Mathematics and its Symbolization 25-37 (1967); Uttam Kharde, *The Symbolic Language of Mathematics*, 1 THE EXPLORER: A MULTIDISCIPLINARY JOURNAL OF RESEARCH 117-118 (2016); and Daniel Silver, *The New Language of Mathematics*, 105 AMERICAN SCIENTIST 364 (2017), available online: https://www.americanscientist.org/article/the-new-language-of-mathematics.

[614] Katrin Fundel, Robert Küffner, and Ralf Zimmer, *RelEx – Relation extraction using dependency parse trees*, 23 BIOINFORMATICS 365 (2006).

[615] Defined as a software that transforms data into structures.

[616] Defined as one or more nouns and their subordinate adjectives. *See* Fundel et. al, *supra* 614.

[617] *Id.* at 366.

Analogously, the approach used in the RelEx paper will be applied to the current analysis of legal judgments. In addition to noun-phrases, sentences are deconstructed into the basic semantic building blocks of the English language;[618] otherwise, subject-verb-object (SVO) triplets. Sentences selected from each judgment are chosen based on their significance to the outcome of the judicial decision. These sentences are subsequently scanned for the presence of SVO triplets. Markers are then assigned to each individual sentence based on equivalency, in order to then form connections between phrases.

Referring back to the aforementioned linguistic models, applying the RelEx method necessarily depends on a preference to dependency syntax and the classical theory of concepts (definitionism). Nevertheless, we argue that the mapping of each SVO component in reference to its neighboring components helps compensate the pitfalls involved with the multiplex nuances of word usage. By working with context, the analysis will extend beyond the realm of prototype theory,[619] which struggles to explain properties arising from context and pragmatic inference.[620] The graphing of the SVO triplets acknowledges context,[621] thereby becoming an integral part of the overall analysis. This method overlaps with ideas addressed in cognitive linguistics, such as the theory-theory of concepts, that heavily relies on role and context. Furthermore, employing sets of meta-concepts, along with graphed contextual relations, provides an analogy of traversing the semantic and pragmatic layers of language.

The case study is, therefore, guided by three key tools: (1) Python; (2) spaCy; and (3) Neo4j. The first is the formal scripting language used to write the translation algorithm. Python was chosen for its known flexibility and general use.[622] Python also adapts in a number of design spaces, namely for tasks that are structural and reflective. spaCy is the chosen open source[623] library for NLP. spaCy is

[618] In other languages, a finite verb can occur without an overt pronominal subject. This is known as the null-subject, or pro-drop, parameter. The English language lends itself especially well to this approach due to the absence of this parameter. Furthermore, English generally does not allow zero copula forms (cf. Russian "я свободен" ("I [am] free")); this is also conducive to verb anchored SVO triplets in the dependency framework.

[619] Rosch and Mervis, *supra* 575.

[620] Jerry Fodor and Ernest Lepore, *The red herring and the pet fish: Why concepts still can't be prototypes*, 58 COGNITION 253 (1996).

[621] Defined here as other surrounding SVO elements.

[622] For further information on Python and developer knowledge, *see* Python, https://www.python.org/doc/.

[623] Open source is defined as software that is available for anyone to inspect, modify, and enhance. spaCy operates under a MIT license. This form of license is a permissive software license with the sole restriction that the original copyright and license notice be included in any future copies of the software. See *What is open source?*,

the primary software used to help parse sentences from legal texts to dependency trees; then to organize the components into categories.

The decision to use spaCy, as opposed to other NLP packages available in Python, is its ease of use, configurability, speed, and existing models pre-trained on a generalized data (e.g. articles, comments, blogs, etc.).[624] While intuitively NLP programs, such as LexNLP, were considered, the current test case poses a different challenge. LexNLP, for example, works with legal texts that are rather structured (i.e. contracts).[625] Therefore, LexNLP is trained at the document and clause level; thereby capable of extracting and classifying clauses as opposed to semantic content. I acknowledge that there are certainly merits to LexNLP. The greatest advantage being its models are pre-trained on U.S. legal texts. Nevertheless, spaCy offers much more functionality and flexibility given the breadth of subject matter found in the training data. By way of analogy, the choice may be akin to choosing between an oyster knife and a Swiss army knife when asked to descale a bass. The oyster knife is specialized but has its practical limits. In contrast, the Swiss army knife – emblematic of versatility – may offer more options and space for creativity when handling intricate tasks.

Finally, Neo4j is a graph database management system designed to store and process data in the form of nodes and relations.[626] The system helps classify the entities and the semantically relevant connections between such entities. Graph databases are commonly used for intermediate representation (IR). Known as the "steppingstone from what the programmer wrote to what the machine understands,"[627] IR is an object-oriented structure that, in its final form, stores all information required to execute a specified program.[628] IRs facilitate translations from natural language to machine code, bridging semantic gaps and behaving as the 'middleman' between syntactic forms. The graph database is also ideal for modelling dependency trees and object-oriented

opensource.com, https://opensource.com/resources/what-open-source. *See* also *The MIT License*, Open Source Initiative, https://opensource.org/licenses/MIT.

[624] For further details, see spaCy's technical documentation, available at: https://github.com/explosion/spacy-models/releases//tag/en_core_web_lg-2.2.5.

[625] *About LexNLP*, LexNLP, https://lexpredict-lexnlp.readthedocs.io/en/latest/about.html.

[626] For further information, *What is a Graph Database?,* Neo4j, https://neo4j.com/developer/graph-database/.

[627] Cliff Click and Michael Paleczny, *A Simple Graph-Based Intermediate Representation*, 1995 ACM SIGPLAN WORKSHOP ON INTERMEDIATE REPRESENTATIONS 35 (1995).

[628] *Id.*

phenomena, such as inheritance. Put together, we attempt to advance the techniques inspired by RelEx for the translation of legal language to numeric form.

b. Risky Business: Case Selection

The initial test cases selected for the POC are not arbitrary. I have strategically chosen cases that all follow a similar premise: what is the meaning of "use" applied to a firearm? Importantly, the cases belong to an alleged lineage, the application of precedent and consistency in legal adjudication.

In 1993, the Supreme Court of the United States (Court) was asked to rule on the definition of "use" in *Smith v. United States*. The petitioner, John Angus Smith, had offered to trade his gun in exchange for cocaine. He was subsequently charged with numerous firearm and drug trafficking offenses. This included using a firearm "during and in relation to" a drug trafficking crime, as stipulated under statute 18 U.S.C.§924(c)(1).[629] The Court held that the trading of a firearm constitutes "use" within the meaning of the statute. There are two remarkable notes to this case. First, the Court interprets the meaning of use rather broadly, particularly applying emphasis on the "everyday meaning and dictionary definitions" of use. Second, the interpretation is placed in the limited context of drug trafficking. The Court shifts away from a dictionary definition and, instead, emphasizes the furtherance of a crime as influential to the use.

In 1995, the Court was again asked to rule on the definition of use in *Bailey v. United States*. Similarly, the petitioners, Bailey and Robinson, were each convicted of drug offenses and of violating, none other than, 18 U.S.C.§924(c)(1).[630] The factual difference is the state of the firearm "during and in relation to" the drug-related offense. The Court was, therefore, asked to determine whether accessibility and proximity to the firearm was indicative of use. The Court held that the statute required "evidence sufficient to show an *active employment* of the firearm by the defendant, a use that makes the firearm an operative factor in relation to the predicate offense."[631] In *Bailey*, the Court then narrows the definition of use by including the element of "active employment." The Court provides a justification for its decision by referring to Smith and noting the ordinary definition of

[629] *Smith v. United States*, 508 U.S. 223 (1993).

[630] *Bailey v. United States*, 516 U.S. 137 (1995).

[631] *Id.*

"use" in the active sense is "to avail oneself of."[632] Strikingly, the act of bartering falls within active employment, even though the gun was exchanged passively.

Coincidentally, a third case – three years later – had arisen, requesting the Court to rule on the definition of use under statute 18 U.S.C.§924(c)(1). However, *Muscarello v. United States* stretched beyond use and, instead, focused on "carries."[633] In *Muscarello*, enforcement officers had found guns in the petitioners' vehicles stored in a locked glove compartment and trunk respectively. The Court was, therefore, asked to determine whether that sufficiently fell within the definition of "carries." The Court ruled that carrying a firearm, in accordance with 18 U.S.C.§924(c)(1), "applies to a person who knowingly possesses and conveys firearms in a vehicle."[634] The Court again invokes "ordinary English," otherwise, basic meaning in dictionaries, to argue that carry is synonymous with conveys. Moreover, the Court again refers to *Smith*, but unlike *Bailey*, directs its reasoning to the *purpose* of the statute.[635] Notably, in all three cases, ordinary meaning was put forth as a dominant line of argumentation. Yet, the argument was always supplanted by intentions of Congress and the statute; that the purpose is to combat the "dangerous combination" of "drugs and guns."[636]

Funnily – perhaps to avoid a fourth case – Congress amended statute 18 U.S.C.§924(c)(1) to include "possess" in tandem with the phrase "in furtherance of any such crime;" thereby, accommodating the outcomes rendered in *Smith*, *Bailey*, and *Muscarello*. This then limited subsequent cases from arriving at the hands of the Court.[637] These cases were, therefore, carefully selected to illustrate that judicial decisions could bear the epistemic flavors of textualism with an underlying subtext of policy. Moreover, their similarity in factual circumstances allow for a stronger test of the underlying mechanisms of judicial reasoning and legal argumentation.

Again, the cases selected are not without limitations. In fact, they were cherry-picked to better demonstrate the subtleties of language and linguistics in law. Equally, I acknowledge that there are

---

[632] *Id.*

[633] As a clarification, 18 U.S.C.§924(c)(1) involves both use and carries a firearm during and in relation to a drug trafficking crime.

[634] *Muscarello v. United States*, 524 U.S. 125 (1998).

[635] *Id.*

[636] *Smith*, *supra* 629 at 240. Also cited in *Muscarello*, *supra* 634.

[637] This is not to say no further cases were brought to courts involving the "use of a firearm" in a drug trafficking crime. This is only applicable to cases before the Supreme Court.

shortcomings to the project: namely, the importance of fact in law. Geoffrey Samuel states, "law arises out of fact."[638] That is, the legal effect of precedent extends so long as the material facts of the case are analogous. The project, however, does not currently account for the facts of the cases. Instead, they focus on the Court's specific arguments on the meaning of "use," accepting the facts as only peripheral to the exercise. The exclusion of facts may be problematic, given their significance to the nature of the common law system.[639] Still, the intentions of the paper are not to replicate judicial reasoning in common law. Fundamentally, the focus of the POC is translation, specifically an experiment to operationalize the migration of legal texts in natural language to algorithmic form.

III. PRELIMINARY OBSERVATIONS

The inherent nature of interdisciplinary projects exposes the gaps between untraversed worlds. Between a data scientist, mathematician, linguist, and jurist, there are primarily two spheres of operation. One is derived from logic, and the other in humanities. Moreover, the disciplines speak different technical languages. Indubitably, there are clashes. Yet, the unifying mission to uncover 'meaning' has raised interesting perspectives on method and interpretation.

Consider the conversation between the linguist and computer scientist. The linguist struggles with a possible SVO markup for open clausal complements. The computer scientist suggests that it would fit 'cleanly' in the code if this were marked in the same manner as a clausal subject. The linguist is bewildered. In dependency linguistics, an open clausal complement is a clause without a subject. A clausal subject, on the other hand, is when a whole clause is itself a subject. What might be problematic with this type of equivalency?

This particular concern was contemplated within the framework of 'nested SVOs.' Complex sentences are composed of several clauses that carry condition and inherence. For example: adverbial phrases or subordinate clauses, that are themselves SVOs, act as modifiers to an overarching (superordinate) SVO. This became problematic when resolving the SVOs with one another; threatening a possible misalignment between their semantic and syntactic representation.

[638] Geoffrey Samuel, *A Short Introduction to the Common Law* 87 (2013).

[639] Early origins of common law regarded it as a *customary* system of law, a body of practices observed by its players. *See* Vicki C. Jackson, *Constitutions as "Living Trees"? Comparative Constitutional Law and Interpretive Metaphors*, 75 FORDHAM L. REV. 921 (2006).

Another fascinating example came about when assessing the difference between the following two sentences:[640]

> "He shot the man with a gun."
> "He shot the man with a telescope."

For the human mind, the role of the object evidently differs between the sentences. In the former, the gun is indicative of the weapon used by the perpetrator. In the latter, the telescope is a qualifier of the victim, drawing a sharper image for the reader. This is owed to the cognitive association[641] between the object to the verb "shoot." But, what happens should the gun qualify "the man" in the first sentence? If so, not only does it change the meaning of the sentence, but, more importantly, it could affect the ultimate charge against the perpetrator. That is, the crime could be a difference between murder, manslaughter, or self-defense. The sentence alone cannot provide this depth of information required. Context and factual circumstances of the event are needed to determine how the sentence should be interpreted.

Interestingly, the data scientist and/or mathematician would approach the question by calculating the cosine similarity between the vector representations (word embeddings) of the verb and the object. Similar to the cognitive association performed in the human mind, the calculation determines the statistical probability[642] of the object appearing with the verb. The higher the frequency of both words co-occurring in the training corpus, the more likely the object is qualifying the verb. The cosine similarity can, therefore, be used as a numeric interpretation of how the object is employed given the verb in the sentence.

A third puzzle came in the form of homographs. Homographs, though identical orthographically, vary in meaning (though often distinguished in pronunciation). How then could a computer distinguish between record as a noun or record as a verb? The computer scientist notes that a distinction in the meta-concepts would resolve the problem. Meta-concepts, or metadata, are the elements outside of the SVO that describe the information being conveyed. This includes in what

[640] It is important to note that the sentences are not taken from the judicial decisions but were conjectured in the process of completing the SVO markup.

[641] More specifically, the realm of psycholinguistics describes this association as top-down processing: the process through which knowledge and experience subconsciously influences interpretation of language. *See* Paul Warren, *Introducing Psycholinguistics* 137 (2013).

[642] This is to reference the Word2Vec approach and transformer-based architectures that actively employ the surrounded words to mathematically derive context.

manner and how the sentence is being expressed. How important then is meta-data to the meaning of sentences?

This was again proposed as a possible resolution when encountering deictic expressions. Deictic words – such as 'this,' 'that,' 'here,' or 'there' – rely almost exclusively on context. Consider the sentence: "At issue *here* is not 'carries' at large, but 'carries a firearm' (emphasis added)."[643] What could 'here' mean? To the jurist, 'here' represented the material facts of the case, but to the linguist, it is a limited reference to the preceding sentences. To the mathematician or computer scientist, the word *here* represents a subjective concept for which a frame of reference and context serve to anchor it in reality.

These observations culminate to a greater question: what exactly constitutes as context? Meaning hinges on the knowledge of a "word by the company it keeps."[644] Should there be multiple interpretations of context, there are seemingly differing methods of arriving at 'meaning.'

At a glance, the SVO markups are products of conversations around these patterns of dependencies within sentences. Decisions were taken on how the sentences should be deconstructed to better articulate the interaction between subjects and objects with their verbs. Equally, an evaluation was made to separate meta-data from the basic SVO structures. Once the SVO markup was complete, it would form part of the training data for a decoder algorithm. The algorithm not only draws out the rules from the markup, but also other rules that the machine has gathered. This theoretically mirrors the concept of "reading between the lines." Finally, these rules are encoded for future documents in the graph created. The idea is that the markups identify only the more pertinent information in each sentence, while the algorithm detects any surrounding information.

The purpose is then to illustrate the connections and changes in the states of sentences found in the judicial decisions. In other words, it is the reconfiguration of sentences that are ostensibly void of structure, to their structurally dependent forms. In the following section, I articulate in detail the technical implementation of the project. I hope to demonstrate that the translation of legal text to numeric form unravels the 'Black Box' of instinct[645] and disciplinary bounds. In the process of

[643] *Muscarello* (Ginsburg, J., dissenting), *supra* 634 at 145.

[644] John Rupert Firth, *The Technique of Semantics*, 34 TRANS. PHILOS. SOC. 36 (1935).

[645] Recall Simon, *supra* 569. *See* also R. George Wright, *The Role of Intuition in Judicial Decision making*, 42 HOUS. L. REV. 1381 (2005); and Chris Guthrie et. al, *Blinking on the Bench: How Judges Decide Cases*, Cornell Law Faculty

reducing sentences to SVO triplets, what is colloquially understood as intuition and knowledge-based expertise is revealed in a systematic form.

IV. TECHNICAL IMPLEMENTATION

As discussed, there have been attempts at translating natural language to numeric form using various types of algorithms. To this day, success has primarily been achieved with the use of advanced statistical modelling techniques that depend on vast amounts of data. Leaning into these methods, we attempt to develop a new paradigm for natural language understanding, namely, one based on the core principles of Object-Oriented Design (OOD). The objective is to develop a preliminary model capable of ingesting a large amount of the data accurately, leaving the handling of outlier cases for a later stage of analysis.

Building on the ideas of Walter Daelemans and Koenraad De Smedt,[646] we refer to their work to bridge concepts of OOD and linguistics. As the intention is not to be exhaustive, the table below broadly defines the analogies between OOD and linguistics that permit the translation of text into this form:

Object-Oriented Design	Concepts from Linguistics
Classes Blueprints (or prototypes) defining the characteristics and behaviors of Objects belonging to them	Hyponymy,[647] items contained in a set. Defining the prototype entities which allow objects to inherit any combination of single or multiple parent characteristics.
Objects Singular manifestations of a Class	Noun-phrases and lexemes corresponding to singular entities and qualities (akin to individual definitions) represented by their lemma-form.
Methods A defined interaction event between Abstractions in the program. Methods must be invoked in order for them to have a role.	Clauses (narrowed down to possible permutations of (S)V(O)) – interactions between semantic entities within the text. The syntactic *subject* (semantic *agent*) is seen as the triggering *entity*, the (direct) *object* is the target of the interaction, and any additional *objects* behave as necessary inputs concerned in triggering the said interaction. The *verb* describes what happens during the interaction.
Variables Placeholders for discrete information: values, Objects	Meronomy (declaring/assigning a placeholder for a part of the whole), meronymy (defining the content in the placeholder) – assigning parts of a whole.

Publications Paper 917 (2007), available at: https://scholarship.law.cornell.edu/cgi/viewcontent.cgi?article=1707&context=facpub.

[646] Walter Daelemans and Koenraad De Smelt, *Default Inheritance in an Object-Oriented Representation of Linguistic Categories*, 41 INT'L J. OF HUMAN COMPUTER STUDIES 149 (1994).

[647] To clarify, hyponymy describes the relationship of 'kind': if A is a type/kind of B, then A is a hyponym. In turn, meronymy is the relationship of 'parts' (also known as partonomy): if A is part of B, it is a meronym. For example, *table* and *chair* are hyponyms of *furniture,* whereas *wheels* and *doors* are meronyms of *car*. *See* Kate Kearns, *Semantics* (2000).

Abstraction The definition of Classes, Objects, Methods and Variables based on the task a program will solve.	Decoupling the signifier from the signified,[648] allowing for the open system nature of language and knowledge in general.
Inheritance The passing on of characteristics and behaviors of a parent Abstraction onto its child	A multi-purpose mechanism allowing the modelling of linguistic phenomena, such as hyponymy, conducive to definitionism.
Encapsulation The localization of characteristics and behaviors to a Class or Object	The phenomenon that allows for semantic parsing – localization of characteristics and behaviors to specific logical elements (entities) within a frame of reference.
Polymorphism The ability to change any inherited characteristics and behaviors	Corresponding to the phenomena of polysemy and homographs, among others. Specifically, it allows any two entities within the same class to have different characteristics and behaviors represented by the same root word.
Composition Arranging the interactions of Objects and Classes with one another; one of the aims of composition is to reduce code **redundancy**	Corresponding broadly to semantics – the arrangement, hierarchy and definition of communicative rules between the logical/semantic elements (perhaps equivalent to semes or sememes) in a text. This can allow for abstraction, improving **efficiency** in contextual assignment.

As such, the grammatical structure of natural language is seminal to extracting its informational content. This would, in effect, permit a translation of 'meaning' to a form readily encodable in a programming language.

The complexity of legal concepts (i.e., the potential for multiplicity of meaning; or polysemy) called for technology that could cater to non-singularity. Consequently, the project attempts to strike a balance between definitionism and determinism by minimizing the pitfalls of both; the inefficiency and redundancy of definitionism against the brittleness of determinism. Ultimately, the goal is to secure efficient machine readability while upholding fundamental legal principles. The danger of leaning towards either the former or the latter is its adverse impact on the requirement for human intervention in the exercise of judicial reasoning. Should priority turn to definitionism, we risk creating a system that is far too complex and cumbersome to create any additional value for legal practitioners. Should priority turn to determinism, we risk creating a system that does not leave sufficient flexibility for ever-changing circumstances;[649] undermining existing legal structures.

[648] Referencing Ferdinand de Saussure and Jacques Derrida on semiotics. *See* Ferdinand de Saussure, *Course in General Linguistics* (Bloomsbury Revelations Ed. 2013); and Jacques Derrida, *Limited Inc.* (1988).

[649] Against the dismay of determinate expert systems, I am cognizant of judgments as temporally specific reflections of society; often, subject to influence by its sociopolitical environment. Notably, disruptions and shifts in society could (and often do) lead to reversal of judicial decisions. *See* for example the commentary by Kiel Brennan-Marquez and Stephen Henderson, *Artificial Intelligence and Role-Reversible Judgment*, 109 J. CRIM. L. & CRIMINOLOGY 137 (2019).

Graph databases are amenable to generating highly interconnected webs of knowledge (knowledge-maps), optimizing analysis of relations between individual data points. Moreover, it accounts for issues of object composition, polymorphism, encapsulation, and inheritance; and enables the use of graph theory for creative analytical approaches on a larger scale. These ideas will return in the subsequent sections. Importantly, the graph works as the intermediary interface. It stores the input and analyzes the output of abstractions drawn from the developed algorithm.

In the normal reading of texts, humans typically abstract in a sequential pattern; forming a 'world' within our own consciousness. Each subsequent phrase that speaks to the same topic enriches the details of this 'world,' reinforcing it with logical constraints and other abstractions.[650] This parallels a compiler reading a piece of high-level code, such as a Python script. The input works through layers of translation before arriving to a form comprehensible to the machine. Each stage serves to 'decompress' the knowledge built into the language by its designers. Eventually, the language is distilled down to its most granular level: a collection of binary code.[651] Phrases become a series of commands; either establishing a fact or describing an event or action.

The legal language is no different. It can be regarded as the sum effort of numerous iterations of layered abstractions rooted in social reality.[652] A legal document is the written manifestation of this process; conveying abstract legal concepts in a manner that is both syntactically sound and semantically meaningful in natural language.

One of the notable pitfalls of natural language is the underlying difference in contextual knowledge, whether it be prior experience or preconditioning. The existence of these differences manifests as "biases," which are then inherited in physical repositories, or artifacts.[653] Consequently, exposing context is often helpful in clarifying such 'repositories of legal knowledge.' For programmers, what is interpretable as context is the workings of reality outside the scope of a particular program. This could mean additional software may be used by developers when putting together a system (e.g. the importing of packages in Python). The addition of these packages extends the functionality of a

[650] Warren, *supra* 641.

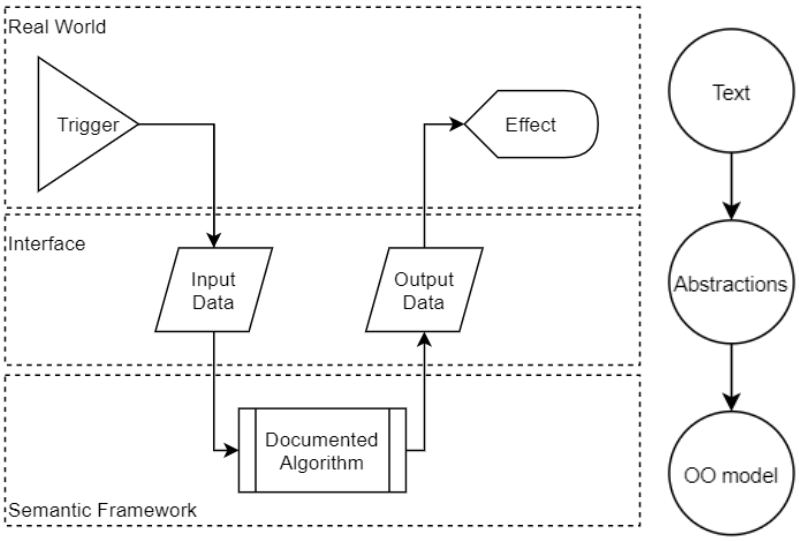
[651] To recall, this is an array of 0s or 1s to control transistors. It is the smallest unit of measure and often regarded in the logic form of an if-then statement.

[652] *See* for example Joseph Raz, *The Institutional Nature of Law*, 38 MODERN L. REV. 489 (1975); also, the difficulty of demarcating legal concepts in Joseph Raz, *Legal Principles and the Limits of Law*, 81 YALE L. J. 823 (1972).

[653] Langdon Winner, *Do Artifacts Have Politics?*, 109 DAEDALUS 121 (1980).

program beyond its defined code. For the POC, we used a combination of pre-defined (i.e. spaCy's neural network models for recognizing dependencies and part-of-speech tags as well as Word2Vec converters) and newly trained estimators (i.e. detecting SVO triplets) to strengthen the model with metadata relevant to statements encountered in the dataset.

Below is a pictographic interpretation of the process:



a. Defining Entities (Encapsulation)

In building reference models of reality, entities are discrete units of existence. They act as mental placeholders to facilitate explanations of interactions within the model. Encapsulation is used to localize the characteristics and behavioral characteristic of each of these entities. The entities can be grouped into categories (classes), nested and (re-)arranged in an infinite number of ways. The importance is the architecture and its rules of performance; in other words, the process of defining entities of reference, their relations to one other, as well as their methods of interaction.

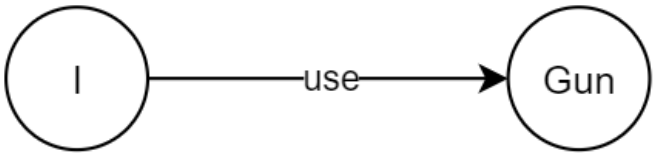
Consider the following sentence from *Bailey* as an informative example:

> "I *use* a gun to protect my house, but I've never had to *use* it."[654]

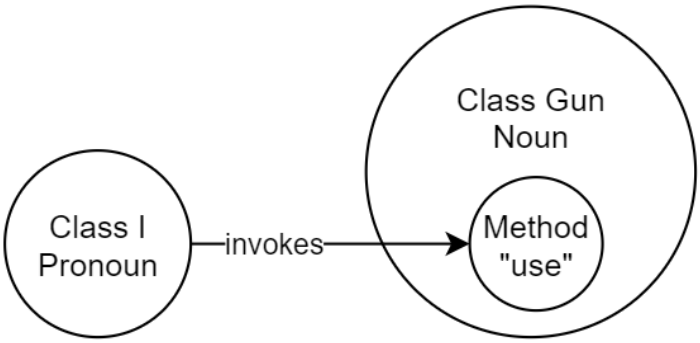
[654] *Bailey*, *supra* 630 at 143.

Disregarding first context, the sentence can be deconstructed into entities or methods. The entities, such as “I”, “gun”, “house,” are encoded as nouns. The methods, such as “protect” and “use,” are encoded as verbs.

Observably, the clause “I use a gun...” involves an actor (“I”) that invokes an action (“use”) on an object (“gun”).



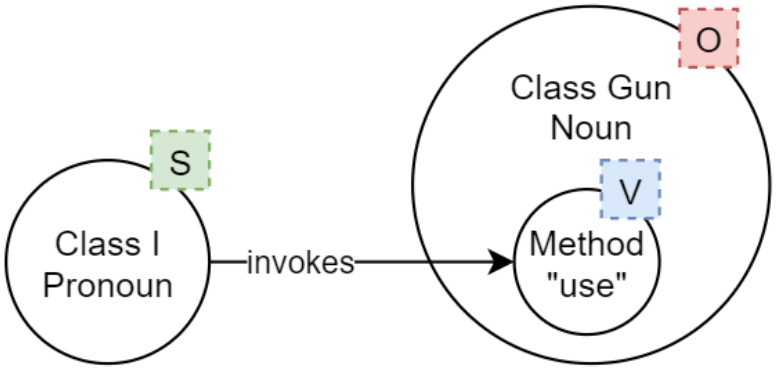
Applying the Object-Oriented approach of structuring code into classes and methods, the first phrase can be translated into the following schema:



The components of the sentence become identifiable SVO triplets:

(1) the Subjects (invoking entity);
(2) the Objects (entity being acted on);
(3) the Verbs (method); and occasionally,
(4) the Prepositional Objects (additional entities describing the premise of the event/action).

The breakdown illustrates the framework on which the algorithm is built.



By extension of the example, subsequent phrases follow a similar breakdown, drawing connections between classes and their corresponding methods. This form of deconstruction also permits the nesting of concepts and additional logic tests along connections established.

b. Scaling Up (Composition)

The process is akin to the first layer of translation, developing a pseudo-code script that represents a concept but expressible in a machine-readable language. The connections trace which class invoked the method "protect" on the class "house;" thereby deducing what "I" "use" to "protect" "house." As a result, such encoding does not require vast amounts of training data. Text is immediately translated to pseudo-code, without the need for external context.

The peripheral terms present in the sentence serve to indicate higher order concepts such as enumeration, negation, time, possession and pronoun assignment: "a", "never", "had to", "my", "it". Their presence exists to modify the fundamental building blocks of the sentence - the nouns and verbs.

c. Creating the Knowledge Map (Natural Language Processing)

Whereas the task of defining individual entities and methods is relatively straight-forward, creating a knowledge-map correspondent of the above schema requires the extraction of the semantic connections between them. By leveraging existing NLP tools,[655] such as spaCy, in conjunction with

[655] I do take note that even the most advanced language parsers are incapable of 100% accuracy. In analyzing the preliminary results, I have encountered a number of deficiencies owed to the dependency trees used. However, at this

our own SVO markups,[656] we were able to create a corpus to train a classifier capable of detecting SVO triplets and importing them to the graph.

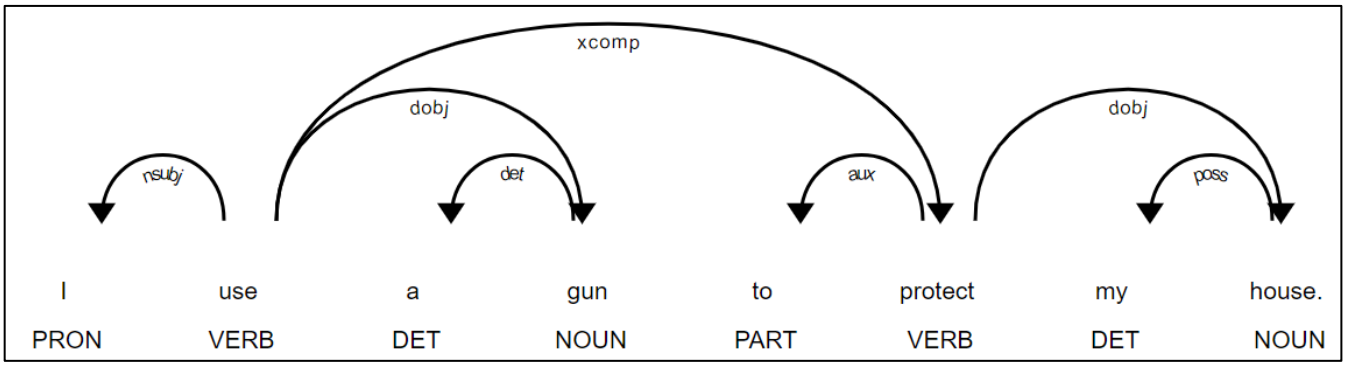


Figure C Sample Input to spaCy

The core strategy behind extracting SVO triplets lies in its linguistic deconstruction. The root of every sentence centers on the verb. Subjects ("nsubj") and objects (predicate, "dobj") are subordinate to verbs within the syntactic hierarchy. Therefore, in identifying the verbs of every sentence, the semantic connections are naturally found.

This method of text analysis has gained popularity with the advent of machine learning based models of NLP; trained on a sizable corpus of different expressions to perform the following tasks:

(a) Separating words from a string;
(b) Grouping the words into sentences;
(c) Assigning each word with a part-of-speech tag (Noun, Verb, Adverb, etc.); and
(d) Estimating each word's syntactic parent; thereby build a syntactic tree

This approach differs against other methods of semantic notation that rely solely on syntax; and less on the underlying pragmatics.[657]

Between entity–method and SVO extraction, the data generated is sufficient to begin assembling together the knowledge-map. More importantly, the aforementioned process is derived entirely from

stage, the aim is again to capture a significant portion of the information within the text and leave outlier situations for the next stage of the project.

[656] Recall the RelEx method described in Section III. *See* Fundel et. al, *supra* 614.

[657] Recall subsection on Linguistic Influences and differences between dependency and constituency-based representations.

the text itself. As a result, a defined cause-and-effect type algorithm is built, executable in full or in part, tested and queried. Additional metadata such as word embeddings, sentiment analysis and recognized named entities can provide supplementary information helpful for optimizing the knowledge-map and achieving a stronger understanding of the semantic content.

d. Building Character; Adding Context (Inheritance and Polymorphism)

The case study considers the transformation of legal texts to an Object-Oriented-like script; effectively using 'pseudo-code' to depict concepts embedded within the text. In natural language, multiplicity of meaning could occur when a single concept applies to several circumstances. Different conclusions can be drawn depending on the characteristics inheritable from a parent class. To clarify, this would include determining whether a "firearm" is within the same class as "gun." Similarly, other characteristics may include the methods or actions (verbs) invoked by a particular class. In object-oriented design, this phenomenon is known as polymorphism.

A core aspect of the translation to object-oriented form, as described in Daelemans and De Smedt's paper, is the assumption that subclasses 'inherit' the characteristics of the parent class by default; unless they are hard-coded otherwise.[658] In this case, characteristics and their behaviors are explicitly stated in the legal text. Consequently, if necessary and provided sufficient examples in the source text, as well as a threshold occurrence ratio, it will be possible to migrate certain characteristics up the inheritance hierarchy. Any such event can be signaled with a flag that the presence of this characteristic is an assumption with X percentage occurrence rate among child objects.

[658] Daelemans and De Smedt, *supra* 646.

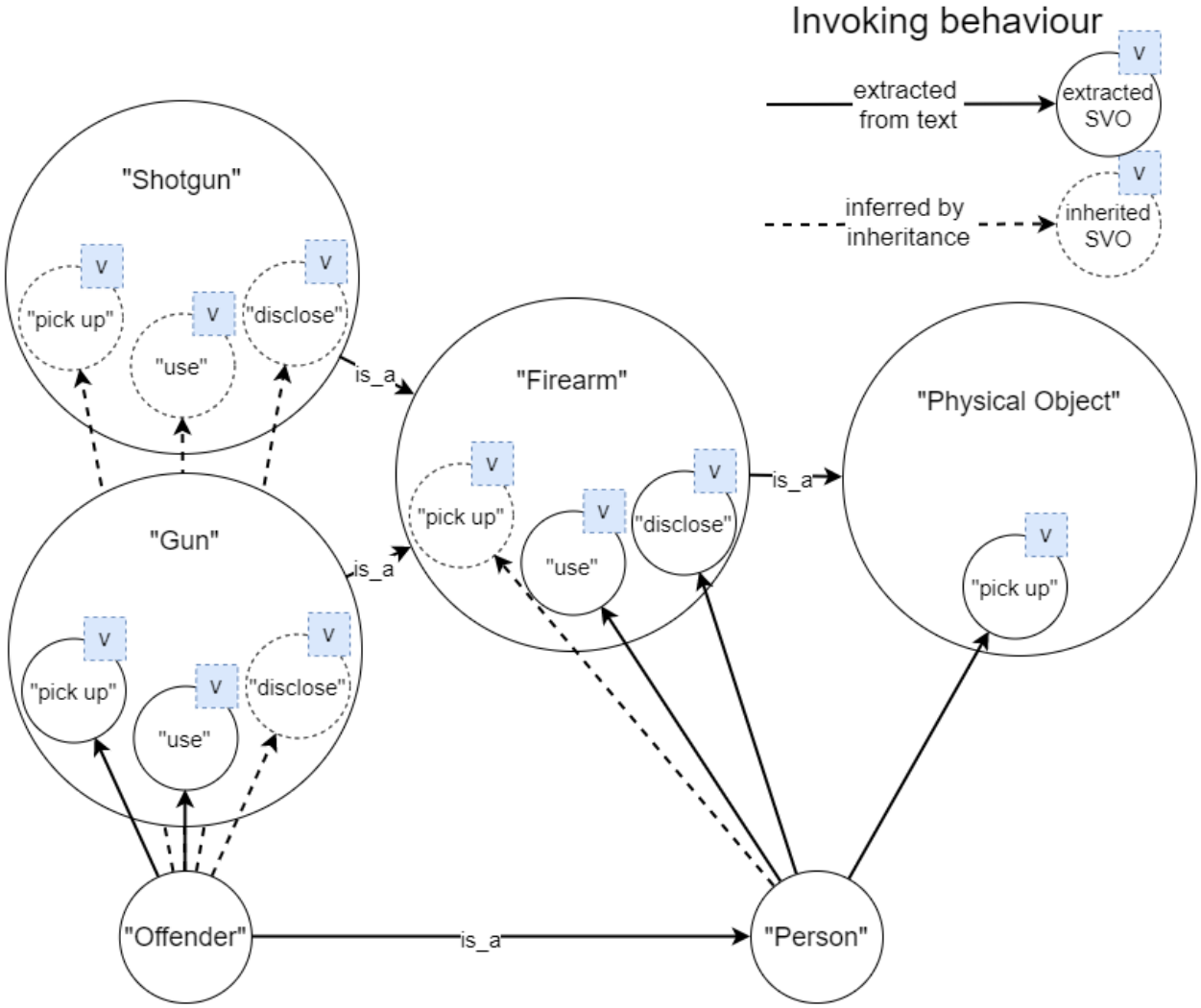


Figure D Illustrating Parent and Child Classes

When SVOs have an explicit subject and object, they can be loosely chained. However, the presence of subordinate clauses in the text necessitates nesting SVOs within one another. This exists in the pseudo-code as implicit causality. To then define the chain of causality, yet maintain the independence of each SVO, the root of a sentence must be identified. Drawing from the example, "I" must first "use a gun" in order to then "protect" "house". This suggests that "use" is the primary connection between the SVOs as one cannot exist without the other.

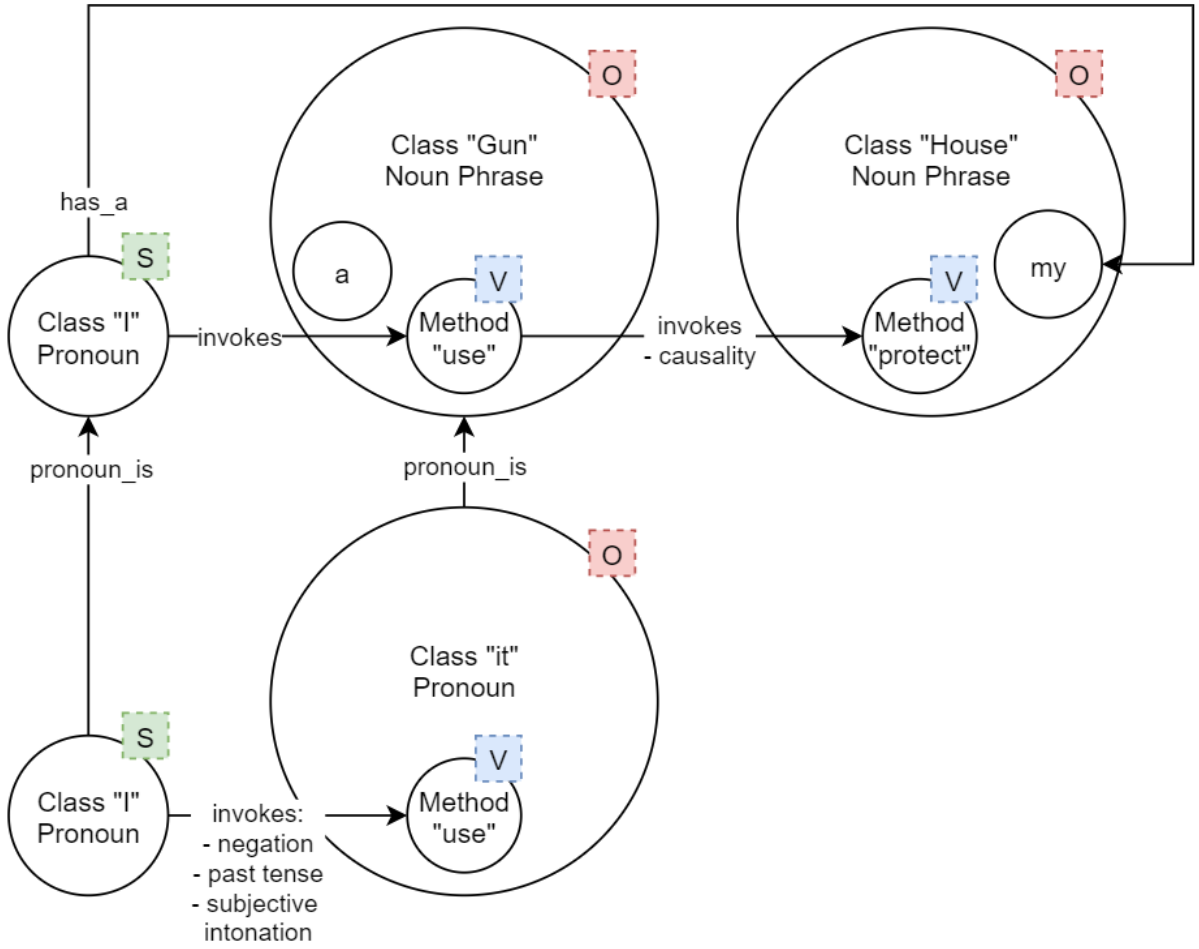


Figure E Illustrating causality between SVOs

Further classifications and qualifying characteristics may be important in a legal analysis. This information parallels the referencing of statutes and case law for prior interpretations of meaning. Various sources of law often create an environment for conflicting readings of a particular text. To tackle this problem, it is possible to assign an authority metric to each source; thereby establishing hierarchical structuring of the corpus. The structure behaves as a type of input when conducting an analysis, mirroring the hierarchy of legal sources.

V. EARLY ACHIEVEMENTS AND FURTHER CONSIDERATIONS

a. By the word of the law

Once the data was loaded into the graph, so began the stage of analysis. The primary way of interacting with the knowledge graph is the query function. Each query attempts to build one or

more paths between two entities, with specific constraints along its path. This is the programming equivalent to writing tests for a piece of code. The knowledge graph is asked a question and returns a response that follows the reasoning of human observers. Once the knowledge graph has acquired sufficient data, the intention is to develop a user interface able to answer 'legal' questions posed by its users.

An invaluable tool used in this task is the Cypher query language. This language permits the formulation of queries based on the paths present within the data. The choice of constraints for each query will initially be hard-coded. Nevertheless, it is possible to then transfer the process to machine learning should sufficient data be gathered.

The idea behind this approach is to shift out of the standard statistically driven paradigm and allow the inference of logical conclusions from the text.

Consider a user query: "Describe the interactions involving a firearm."

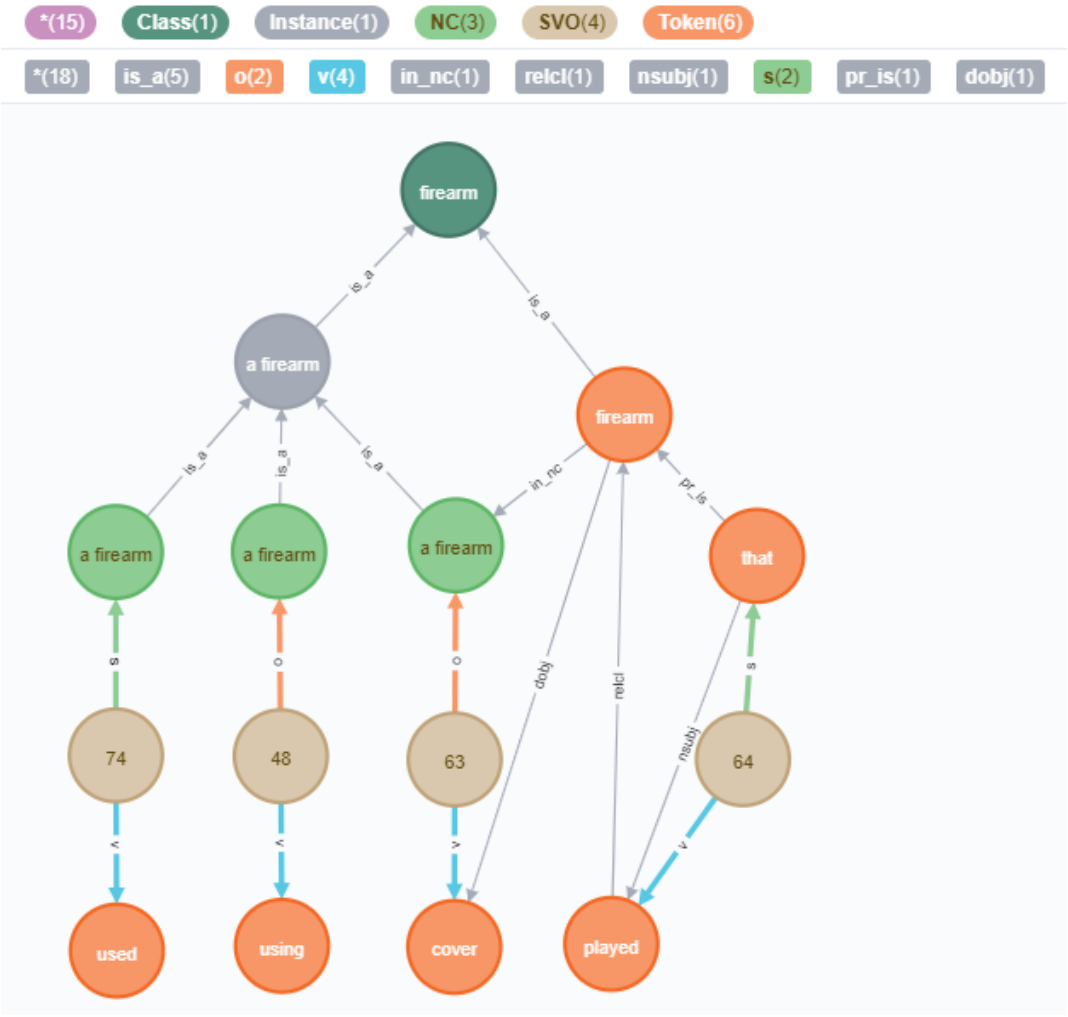


Figure F Sample Output from Neo4j Graph

With a user interface, we envisage that any question will be deconstructed in the same way as the training dataset. In this case, the algorithm should return the associations of entities and methods affiliated with "use" and "firearm." The interface will attempt to: (1) link the entities in the question, using the data in the graph; (2) gather any conditions and constraints along the way; and (3) return the relevant information as a series of possible paths taken within the graph, resulting in a list of phrases sorted by relevance (e.g. "use is active employment").[659] In effect, legal judgments are reconfigured into machine readable form to identify the meaning of the text. The graph acts to signpost legal actors towards definitions found in judicial decisions; thereby augmenting legal reasoning by leveraging the efficiency and power of computational analysis.

[659] *Bailey, supra* 630 at 137.

b. By the sixth sense

On the other hand, there has been a latent understanding that intuition plays a role in the rendering of judicial decisions.[660] The techniques used in our approach, in fact, account for instinct. The parsing of legal texts requires two types of algorithmic methods: (1) analytical; (2) and numerical.

The former serves to build a rigid structure from text and establish a hierarchy of semantic content on the basis of clearly defined criteria. This was demonstrable in the use of the graph database. The latter leverages the statistical modelling principles of neural networks. Similar to impulses attributable to intuition, the weight of each neuron in a neural network can be viewed as an abstract meta-concept; too complex to express tangibly. A parallel can be drawn between the phenomenon of a "gut feeling" to the internals of a neural network, as trends embedded within a dataset are sorted into an array of codependent activation values. This means that any data present on the graph can be fed to customized machine learning algorithms to approximate human 'intuition.' Together, we could factor several forms of legal reasoning that often underlie judicial decisions.

c. Between implementation and effect

To come full circle, the impact of translation has inadvertently exposed the logic of legal reasoning. Whether it is judicial intuition or syllogistic application, Holmes's paradox remains relevant. Words of legal text do, in fact, intrinsically embody meaning. The sphere of legal knowledge exists well within the sentences of judicial decisions. This is owed to the interpretation and conceptualization of precedent. The POC has observed that the use of precedent is not a procedural legal tool but a substantive one. Its application is to uphold the appearance of methodological consistency within the body of law. Yet, fundamentally, its use is to substantiate the authority of legal texts.

More importantly, precedent recognizably functions in an asymmetrical, as opposed to syllogistic, manner.[661] To recall, *Bailey* does not apply the plain meaning of 'active employment,' but constructs instead an alternative legal meaning to equate 'active' as "operative factor."[662] In other words, in accordance with *Smith* and *Bailey*, the use of a firearm includes bartering; and as such, the trading

[660] Recall discussion on intuition in judicial decision making; *see* Wright and Guthrie, *supra* 645.

[661] Countering Holmes's description of the law as following syllogistically from existing precedents. *See* Holmes, *supra* 567.

[662] *Bailey*, *supra* 630 at 143.

of a firearm is an 'operative' component to a drug-trafficking crime. These definitions are not logically deduced. Instead, they seek to reinforce a specific legal framing. Arguably, then, the use of precedent is not to follow past decisions, but to determine how to align with them. This was integral to incorporate in the graph, as the semantic content drew from legal taxonomy.

The result of translating legal text in the manner described in Part IV corroborates that legal language is self-referential and consistent. The law pushes outward by looking inward. In deconstructing legal judgments to its constituent components, the process of applying precedent evidently evolves: from syllogistic application to a framework of extraction.

CONCLUDING REMARKS AND NEXT STEPS

The fundamental question asked by the project is whether meaning draws association from the language in which it is seated; that in changing the language, meaning will naturally be reconceptualized. The test to translate natural language to numeric form is not novel. In fact, it follows an ancestry of applying mathematical precision to legal expression. This case study has sought to experiment with the conversion of legal texts into algorithmic form. More importantly, I attempted to capture legal concepts and processes involved in legal reasoning. The deconstruction of natural language phrases to SVOs atomized sentences to their bare structures; forcibly exposing connections integral to the formation of concepts. As I aimed to reconcile syntax with semantics, structure became indistinguishable from content.

Inadvertently, the POC has demonstrated that, though form is seminal to the adjudicative exercise, the logic embedded within legal texts does not necessarily behave syllogistically. Instead, legal concepts appear to evolve sporadically. This sporadicity, however, is not synonymous to randomness. Rather, the development of the law draws from introspection and uses precedent to substantiate its authority. Teasing at Holmes's paradox, the law approaches consistency not in form, but in substance. As opposed to syllogistic application, meaning is found through a process of extraction.

Beyond the case study, the next phase of the project intends to bring forth a deeper breakdown of legal texts, focusing on higher levels of abstraction (i.e., trends latent in meta-concepts) and more complex grammatical resolutions found in natural language. From a broader perspective, the case study has inspired us to consider advancing towards a 'White Box' solution. The aim is to strengthen

the understanding of legal texts, providing richer roadmaps and signposting users towards more consistent interpretations of judicial decisions. It is an evolution of legal reasoning that heightens transparency by unpacking juridical truths and structuring intangible legal narratives. The result? Improving the quality of legal analysis and elevating accessibility to society.

As opposed to "grafting new technology onto old working practices,"[663] it is a new embodiment of precedent. It is a harnessing of the future through a preservation of the past. The integration of computational technology in law disrupts conventional legal mechanics, while maintaining the function of law. I anticipate then a Bilbao effect, that the thoughtful marriage of old and new architecture sparks transformation.

[663] Referencing the distinction Susskind makes between automation and transformation. *See* Richard Susskind, *Online Courts and the Future of Justice* 34 (2019).

3C- The Legislative Recipe (Machine-Readable Legislation)

I have noted (and perhaps stressed) that legal interpretation is, in part, a linguistic venture. As notable in judicial opinions, courts are often asked to interpret the text of statutes and legislation. The question becomes: what if there was a method of extracting the meaning of statutes consistently? This is the fundamental basis of the Rules as Code initiative. That is, encoding legislation in a mathematically precise form would permit clearer responses to legal questions.

To recall, Layman E. Allen lamented about ambiguity in legal drafting owed to syntactic uncertainties.[664] In his fascinating study, he deconstructs an American patent statute and notices immediately the complexity with the word 'unless.' He asks whether the inclusion of 'unless' asserts a unidirectional or a bidirectional condition.[665] That is, does the clause mean (a) if not x then y; or (b) if not x then y **and** if x then not y?

Though nuanced, Allen exposes an ambiguity that muddies the legal force of the statute. An interpretation of 'unless' as a bidirectional condition raises the question of what 'not y' would mean. In this particular case, this could affect whether exceptions are possible in determining patent eligibility. In short, for Allen, legislative language must have a clear structure.

This case study attempts to unpack the notion of machine-readability, providing an overview of both its historical and recent developments. The case study will reflect on logical syntax and symbolic language to assess the capacity and limits of representing legal knowledge. In doing so, the paper seeks to move beyond existing literature to discuss the implications of various approaches to machine-readable legislation. Importantly, this study hopes to highlight the challenges encountered in this burgeoning ecosystem of machine-readable legislation against existing human-readable counterparts.

A. Historical Roots: Symbolic Logic

The code of Hammurabi[666] is frequently used as an example of how the law has changed in form in order to improve access to the legal system, lead to more predictable legal outcomes, and to promote transparency. Through the adoption of form, law can be understood as a body of knowledge that

[664] Layman E. Allen, "Language, Law, and Logic: Plain Legal Drafting for the Electronic Age," B. Niblett (ed.) *Computer Science and Law* 76 (1980).

[665] *Id.* at 77.

[666] Michael Genesereth, "The Legacy of Hammurabi" (Mar. 17, 2021), available at: https://law.stanford.edu/2021/03/17/the-legacy-of-hammurabi/.

over time has come to inform behavior through the production, dissemination, and evaluation of the rules. Lawrence Lessig and Alex "Sandy" Pentland each have highlighted this with the notions that code is law, and law is an algorithm.

These ideas are not new. As discussed in prior case studies, this ancestry dates back to twelfth century logicians reflecting on the use mathematically precise forms of writing. In the mid-1930s, German philosopher, Rudolf Carnap, reflected on a logical syntax for language.[667] His argument is that logic may be revealed through the syntactic structure of sentences. He suggests that the imperfections of natural language point instead to an artificially constructed symbolic language to enable increased precision. Simply put, it is treating language as a calculus.[668]

In this perspective, there is no consideration of language for the intentions of meaning and interpretation. Merely, logical syntax is concerned with structure and is void of content.[669] Though Carnap concedes that syntax belongs to the scientific study of language that enables mathematical calculation, this approach must be distinguished from semantics, or semasiology. For Carnap, syntax importantly builds a system of reference. In an analogy with the "complicated configurations of mountain chains, rivers, frontiers, and the like," geographical coordinates are mathematical constructions that act as informative measurements of comparison to reveal and analyze the behaviors of its 'natural' existence.[670] Symbolic language, therefore, acts to investigate and identify consistencies and contradictions in language for the purpose of clarifying its logical properties.

Since the 1950s, Allen had argued for the inclusion of symbolic logic to develop a systematic method of drafting. The transformation of an ordinary statement to a "systematically pulverized form"[671] would lead to specific and unambiguous legal expressions. Allen's technique is suggestive of two key thoughts: all statements are (1) composed of constituent elements; and (2) built on logical relationships.

---

[667] Rudolf Carnap, *Logical Syntax of Language* 2 (Routledge English ed. reprint, 2014).

[668] *Id.* at 4.

[669] *Id.* at 7.

[670] *Id.* at 8.

[671] Layman E. Allen, *Symbolic Logic: A Razor-Edged Tool for Drafting and Interpreting Legal Documents*, 66 Yale L. J. 833, 835 (1957).

He uses implication/co-implication ambiguity[672] to illustrate how symbolic logic could clarify legal imprecision. He considers the conditions for when a seller may rescind a contract or sale as an informative example. Breaking down section 65 of the Uniform Sales Act into six constituent components,[673] Allen argues that even a "relatively simple and straightforward statutory passage…often [has] a wide variety of possible interpretations."[674] For the specific case of section 65, he found that there are eight interpretations a court could take.[675] Yet, of the eight, only one interpretation tends to be adopted by courts, owed to the contextual support of other sections of the statute.

Allen suggests, by systematically pulverizing statements of the statute, clearer intentions may be revealed. This method acts as a tool to counter drafting in a "broad and ambiguous form."[676] To recall, Stephen Wolfram made a similar argument. Simplification, he states, occurs through the formulation of a symbolic discourse language. If the "poetry" of natural language could be "crushed" out, one could arrive at legal language that is entirely precise.[677]

Machine-readability[678] appears then to bridge the desire for precision with the inherent logic and ruleness[679] of certain aspects of the law. Machine-consumable legislation may, therefore, be regarded as a product that evolved out of the relationship between syntax, structure, and interpretation. In other words, a potential recipe to resolve the complexity of legalese. What Allen intentionally evades, and is rather significant, is the difference between semantic and syntactic uncertainty. While syntactic uncertainties are often inadvertent, semantic uncertainties are often deliberate. The distinction between syntactic with semantic uncertainty is a mirror to unintentional and intentional ambiguity.

---

[672] Defined as whether the connection between two elements of a statement is conditional or biconditional. *See* id. at 855.

[673] *Id.*

[674] *Id.* at 857.

[675] Allen conducts a simple mathematical calculation around the number of interpretations. He notes that where the number of antecedents (otherwise, conditional statements) in the statement is equivalent to N, the number of possible interpretations is equivalent to 2^N. See *id.*

[676] *Id.*

[677] Stephen Wolfram, "Computational Law, Symbolic Discourse, and the AI Constitution," Ed Walters (ed.), *Data-Driven Law: Data Analytics and New Legal Services* 109 (2019).

[678] While there are distinctions in literature between machine-readable and machine-consumable, I use them interchangeably and treats them as synonymous.

[679] Alluding to the quality described in Frederick Schauer, "Ruleness," Dupret Baudouin et al. (eds.) *Legal Rules in Practice* (2021 Forthcoming).

This act of categorization implies the capacity to delineate within natural language core tenets of ambiguity.

Therefore, the correlative association between unintentional ambiguity and syntactic uncertainty is an audacious claim that innately reduces the challenges of legislative drafting to a symbolic fix. For now, it appears there may be a stronger argument that symbolic logic is better suited as a metric to assess clarity and precision in legal drafting.

B. Plain English Legalese

Symptoms of simplification – efforts to make text more digestible – frequently emerge and re-emerge, working through cycles of fashion in the legal industry. To recall, in the 1960s, David Mellinkoff described the absurdity of the legal language bearing characteristics distinct from common speech. Mellinkoff argues that while there is overlap between the two, the language of the law frequently includes common words with uncommon meanings, use of words and expressions with flexible meanings, and "attempts at extreme precision of expression."[680] Perhaps the most interesting is Mellinkoff's sly remarks at the legal language's valiant yet unsuccessful efforts with precision. He notes the contrast between the plays on meaning against the sharp boundaries around the vocabulary. In defense of precision, the arguments often invoked by lawyers is of clarity; that the wording is justified in making the meaning clearer.[681] The cult around precision in law's language has built a fortress around change, projecting a fear that use of plain language would disrupt the clarity associated with legal language.

Therefore, Mellinkoff seeks to debunk this myth of precision; the elusive "exact meaning," desired by lawyers, that keeps the technical language afloat. Alternatively, he finds that the tools used in the legal community do not reflect precision. First, agreement on what is necessarily precise has never been reached.[682] Precision is occasionally defined as being exact or "exactly-the-same-way." The former alludes to a definite term, whereas the latter points at the mechanism of analogy and application of precedent. In either scenario, Mellinkoff finds issue with the understanding of precision. A focus on definite meaning is misleading as legal language often includes vocabulary such

[680] David Mellinkoff, *The Language of the Law* 11 (1963).

[681] *Id.* at 292.

[682] Mellinkoff describes this as "the choice of 'precise' language goes by default – without notice that any problem exists." See *id.* at 297.

as "reasonable," or "substantial" that are fundamentally imprecise. From the perspective of precedent and argument for tradition, Mellinkoff suggests that precision is merely an effect produced by law's formulas. That is, "an inflexible primitive insistence on word-for-word repetition could make the traditional the precise."[683] Embedded into the legal language is an attachment to form as opposed to meaning. Consequently, the arguments towards precision are, in fact, structural and not linguistic.

Peter Tiersma, decades later, discussed the extent to which legal language was effective as a means of communication. His conclusion was that the goals of the language did not serve the intentions of the law. That is, the desire to appear objective and authoritative conflicted with the use of language in law. Tiersma suggests that legal language has come to be understood as a method of exclusion, an indicator that one belongs to a "legal fraternity."[684] This incongruency enables a continued dependence on the legal community to decipher and translate legal texts.

Tiersma highlights two elements that have worked against the use of plain English in law: (1) the "quest for precision" in law; and (2) the legal lexicon. The former acts as a shield against ordinary English, and the latter is to distinguish law from other disciplines. Perhaps ironically, Tiersma observes that the arguments for legal language – clarity, conciseness, and precision – are also the causes of imprecision and lack of clarity. Like Mellinkoff, he argues that the legal language strategically plays on imprecision, flexibility, and generality of use, as well as a specific vocabulary that is largely arcane and jargon.[685] Moreover, interpretation plays a different function in legal than in ordinary language. Tiersma suggests that in ordinary English, interpretation is focused on the speaker's meaning. In legal interpretation, it is fundamentally a semantic exercise reinforced by the aforementioned lexicon. The differences in the practice of language and the reasons behind their use, in effect, lead to complications surrounding the inclusion of plain English in law. Consequently, decades of effort in converting complex legal language to plain English have been met with minimal success.[686]

---

[683] *Id.* at 299.

[684] Peter Tiersma, *Legal Language* (1999), available at: http://languageandlaw.org/LEGALLANG/LEGALLANG.HTM

[685] *Id.*

[686] In addition to the ongoing dialogue towards a 'plain legal English,' it is perhaps best summarized by William Pitt on the elusiveness and illusion of achieving this conversion. *See* William Pitt, "Fighting Legalese with Digital, Personalized Contracts," *Harvard Business Review* (February 27, 2019), https://hbr.org/2019/02/fighting-legalese-with-digital-personalized-contracts.

Nevertheless, there have been strong efforts of developing a plain English for the legal community. Richard C. Wydick, inspired by Mellinkoff, addresses the design problem raised by Tiersma. The underlying argument is that "good legal writing is plain English."[687] Wydick suggests that distinguishing a legal from ordinary language hinders, rather than promotes, legal work. Furthermore, he contends that there are several quick fixes to translating existing legal to plain language. In his text, Wydick identifies issues of legal language as semantic ones of choice and arrangement. The central discussion is on word use and how to manipulate them "with care."[688] Grammar is equally relevant; to consider foremost the active voice and punctuation.

There have been examples of Wydick's suggestions in practice. The Plain English Movement[689] reflected an eager intent to increase the accessibility of legal knowledge to those outside of the legal community. This was owed to the rising demand for important consumer documents to be made understandable to the general population.[690] Similarly, this has permeated into calls for plain English legislation. Guidelines of 'good faith' were written for legislation to use active verbs and short sentences and be capable of passing the Flesch test.[691]

Despite the vast improvements to the language of consumer documents, most legal documents continued to be written in legalese. If the shift from legal to plain English is as simple and intuitive as described by Wydick, the question becomes: why have the peculiarities of legal language and drafting, persisted? In line with Tiersma's suggestion, perhaps it may be a result of exclusivity. That is, the complexity of the language fosters a continual reliance on the legal community, reinforcing the need for a knowledge translator. On the other hand, there may be a more subtle reason for the preservation of legalese. This argument draws from Mellinkoff's discussion of tradition. Provided that legal language has always been housed in a particular form, there rests an underlying hesitation that legal concepts cannot be expressed in another way. Though Mellinkoff ascribes this to the illusion of precision, it may in fact be an inability to reconceptualize the law. This would again imply

[687] Richard C. Wydick, *Plain English for Lawyers* (2005).

[688] *Id.*, see importantly chapters 6 and 7.

[689] This began with revisions around promissory notes introduced by Citibank in the 1970s. *See* Tiersma, supra 684.

[690] *Id.*

[691] This was considered a "readability" assessment, as it measures the average length of sentences and words. It was suggested that this acted as an objective and quantifiable measurement for comprehensibility. See *id.*

a marriage to the form. In this case, enabling machine-readability would demand perpetuating existing forms of legal expression.

C. Why don't we layer it? XML in Law

From plain English, there took a technical turn. In hopes of developing a better understanding of legislative documents, LegalXML and LegalDocumentXML, products of OASIS Open,[692] were created to provide a common legal document standard "for their interchange between institutions anywhere in the world and for the creation of a common data and metadata model that allows experience, expertise, and tools to be shared and extended."[693] This standard-based approach focuses on assessing the ways in which machine-readable information may be integrated into the official text of legislative documents.[694]

For a document to be made machine-readable, a descriptive markup meta-language,[695] like eXtensible Markup Language (XML), must be embedded into the text in order for a computer to understand it. That is, the document must be deconstructed and sorted into components based on structure and semantics. Structure is defined as the organization and categorization of various parts of the document on the basis of functionality.[696] Semantics, on the other hand, is defined as the meaning, or what the information within the document represents. The intention, then, of decomposing documents into respective structural and semantic framings enables developing a taxonomy and ontology around organizing legislative information.

In effect, standardization is an argument for drawing out and weaving similarities between legislative documents across various jurisdictions. The aim is to increase accessibility and fortify interoperability within the legal ecosystem.[697] As opposed to the existing ad-hoc, or piecemeal, method, the

[692] OASIS Open (accessed Jun. 12, 2021), https://www.oasis-open.org/.

[693] OASIS LegalDocumentML (accessed Jun. 12, 2021), https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=legaldocml

[694] Fabio Vitali, "A Standard-Based Approach for the Management of Legislative Documents," Giovanni Sartor et. al (eds), *Legislative XML for the Semantic Web* (2011).

[695] A form of language used in web programming to allow users to identify individual elements of a document. See lecture slides, "Web Programming," https://home.adelphi.edu/~siegfried/cs390/390l6.pdf.

[696] Vitali, *supra* 694 at 39.

[697] *Id.* at 38-42.

application of a standard technique would encourage transparency in the production and dissemination of legislative information.

As an initial response to a United Nations project to strengthen information systems in legislatures in Africa, a set of standards and guidelines for digital Parliament services, known as the Architecture for Knowledge-Oriented Management of Any Normative Texts using Open Standards and Ontologies (Akoma-Ntoso), was developed.[698] This framework sought to manage information and recommend technical policies and specifications for building Parliament information systems.[699] The results of Akoma-Ntoso led to the three key achievements: (1) the Akoma-Ntoso XML schema; (2) a labelling convention for legal resources (URI); and (3) Legislative Drafting Guidelines.[700] These achievements reflect the broader vision on the use of XML to provide a stronger structural and semantic framework around organizing parliamentary and legislative information. The Akoma-Ntoso XML schema (Akoma-Ntoso), in particular, enables the inclusion of descriptive structure to the content of legislative documents; and, thereby, providing context to legislative information.[701]

The Akoma-Ntoso architecture has been revered as the bedrock on which LegalXML is built.[702] There are two key principles that are fundamental to the schema: (1) descriptiveness; and (2) prescriptiveness. The former emphasizes the preservation of the original "descriptiveness" of the document. This suggests that there is no loss in the integrity of the legislative document, specifically qualitative components that provide important legal or regulatory context. The latter focuses on the implementation of rules, "directly drawn from the legal domain."[703] Together, these principles imply and, perhaps, reaffirm the notion that it may be possible to sort within legal documents elements that are inherently executable and structured; and others that require the detail and particularity. More importantly, Akoma-Ntoso places a focus on the representation and validity of legal documents.[704] The design purports to place at the forefront a proper reflection of legal concepts.

---

[698] Monica Palmirani and Fabio Vitali, "Akoma-Ntoso for Legal Documents," Giovanni Sartor et. al (eds.), *Legislative XML for the Semantic Web* (2011).

[699] *Id.* at 75.

[700] *Id.*

[701] *Id.* at 76.

[702] *Id.*

[703] *Id.* at 77.

[704] *Id.*

Monica Palmirani and Fabio Vitali describe four generations of LegalXML, with Akoma-Ntoso understood as the third generation.[705] Though the differences between generations is primarily based on nuances of structuring, the third generation onward relies on a thorough understanding of object-oriented design.[706] That is, an assessment of patterns and classifications are coupled with an analysis of the relationships between text, structure, and metadata. This process is central to the schema and translation of legal concepts.

In effect, the third generation establishes the "complex multilayered information architecture"[707] that decomposes the legal document from pure text to structured analysis. This multilevel construction is described as a semantic web layer cake.[708] Modelling the document into layers, text and structure, metadata and ontology, aligns again with the implied argument that the content of legislative documents are innately categorical. That is, as opposed to a reconfiguration, or a reframing, of the document, it is instead a question of rearrangement and extraction of these structured elements.

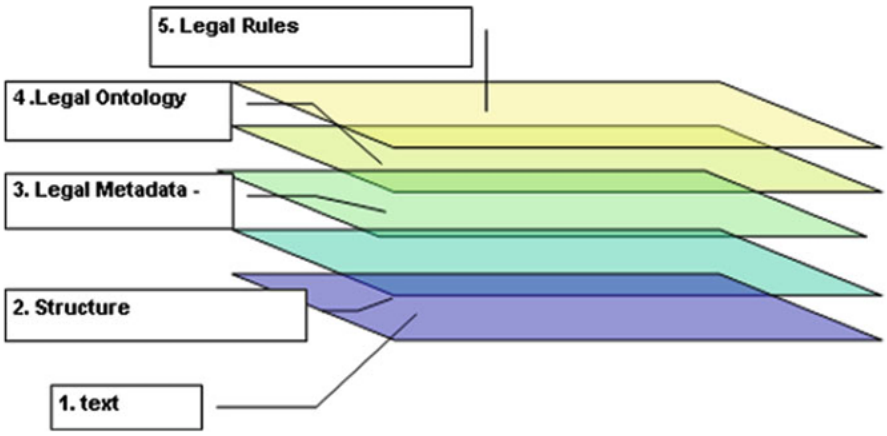


Fig. 6.1 Layers of representation in Legal Document Modelling

How then does LegalXML work? Below are examples[709] of how the layers are drafted in Akoma-Ntoso XML schema and how the relationships between these layers operate. Beginning with the text and structure layers, both layers take from the original natural language and annotate each element semantically. As notable in the examples, the text and structural markup (denoted by these

[705] *Id.* at 78.

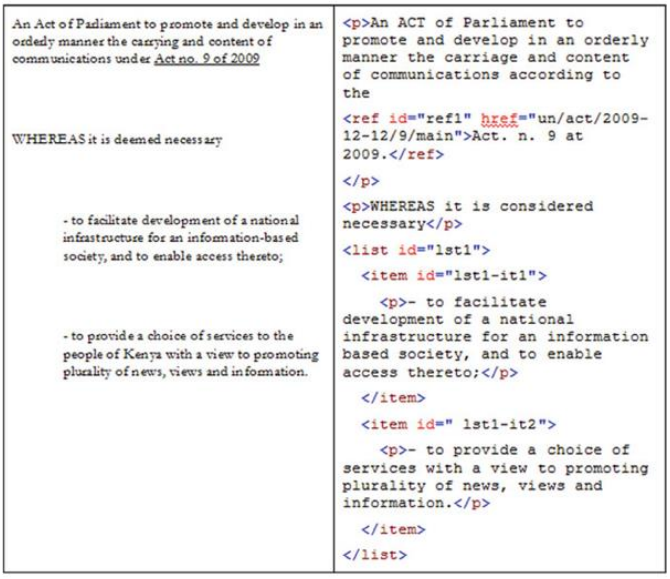
[706] *See* for example in second case study.

[707] Palmirani and Vitali, *supra* 698 at 78.

[708] *Id.* 79.

[709] All examples are taken directly from Palmirani and Vitali's demonstration in their article.

parameters </>), indicate to the machine how the document is organized. Textually, it highlights between paragraphs and references. Structurally, it highlights headers, sections, and subsections.



An Act of Parliament to promote and develop in an orderly manner the carrying and content of communications under Act no. 9 of 2009 WHEREAS it is deemed necessary - to facilitate development of a national infrastructure for an information-based society, and to enable access thereto; - to provide a choice of services to the people of Kenya with a view to promoting plurality of news, views and information.	An ACT of Parliament to promote and develop in an orderly manner the carriage and content of communications according to the <ref id="ref1" href="un/act/2009-12-12/9/main">Act. n. 9 at 2009.</ref> WHEREAS it is considered necessary <list id="lst1"> <item id="lst1-it1"> - to facilitate development of a national infrastructure for an information based society, and to enable access thereto; </item> <item id=" lst1-it2"> - to provide a choice of services with a view to promoting plurality of news, views and information. </item> </list>

Fig. 6.2 Example of text markup

PART I. PRELIMINARY Short title 1. This Act may be cited as the "First Example Act."	<body> <part id="prtI"> <heading>PART I PRELIMINARY</heading> <section id="sec1"> <heading>Short title</heading> <subsection id="sec1-sub1"> <content> 1. This <ref id="ref12" href="/un/act/2010-01-01/1/main">Act</ref> may be cited as the First Example Act. </content> </subsection> </section>

Fig. 6.3 Example of structure markup in Akoma-Ntoso

At the metadata layer, annotations become more complex. As opposed to indicating a legislative document's logical connectors and organization, metadata represents the interpretation and context of the document. In the example below, the left panel of the screen represents a textual markup of a particular section of legislation. The right panel reveals the underlying possibility for multiple interpretations of this section. Therefore, the <mod id=mod1> denotes that for this specific case, there may be two equally valid interpretations: (1) authentic; or (2) exception.[710]

[710] *Id.* at 82.

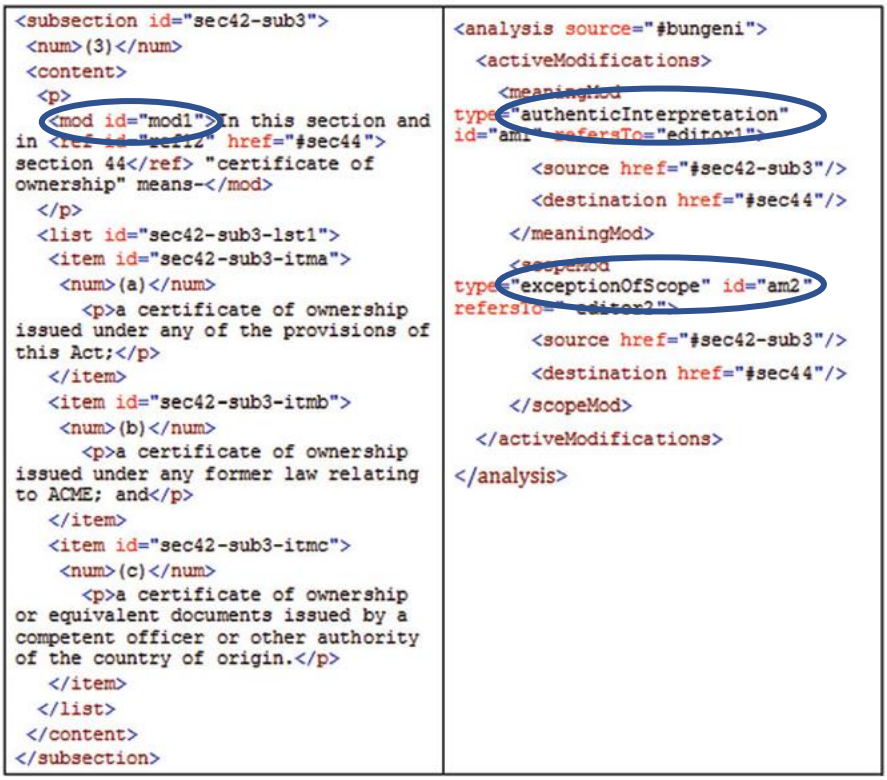


Fig. 6.4 Example of metadata markup connected to the structured text

Moreover, metadata annotations clarify the "local" meaning.[711] For reasons of simplification and uniformity across categorization, Akoma-Ntoso intentionally uses a single convention for all documents. This enables a "shared conceptual architecture"[712] across the legal ecosystem. Therefore, to avoid confusion, the metadata annotates the specific meaning at hand. Below, the docProponent refers to the source of authority. In the left panel, the legislation indicates the legal authority draws from the Ministry of Local Government. The right panel indicates the source draws from the Supreme Court of Appeal.

[711] *Id.*

[712] *Id.*

```
<preface>
  <p class="heading">REPUBLIC OF
AMCE</p>
  <p class="subheading">
   <docTitle> GOVERNANCE FRAMEWORK
BILL</docTitle>
  </p>
  <p class="subheading">
    <docProponent>MINISTER FOR LOCAL
GOVERNMENT of ACME
    </docProponent>)
  </p>
</preface>
```

```
<header>
  <p>
  <b><docProponent refersTo="#SCOA">
SUPREME COURT OF APPEAL
ACME</docProponent></b>
  </p>
</header>
```

Fig. 6.5 Example of shared elements with different semantic meanings

Equally, this shared vocabulary behaves as a legal ontology. It indicates how components of legislative documents belong to broader categories within a legal ecosystem. In the aforementioned, the metadata annotations reveal how a particular piece of legislation connects with other legal documents. More importantly, it localizes where specific interpretations are drawn. This substantiates a more explicit approach on the gathering and understanding of legal knowledge.

Akoma-Ntoso then fulfills the desires of logicians for a legal language that is sufficiently precise. Returning to Allen, if legislation should have a clear structure, Akoma-Ntoso appears as an ideal option. Yet, the rate of its adoption has been strikingly low.[713] This is perhaps owed to the two-fold complexity of migrating legislative documents from text to XML and the requirement of XML competency in the translation process. First, converting legislation from natural language to an XML schema is described as an eight-step recipe.[714] Importantly, it requires first a legal analysis that is typically done on paper. As described by Palmirani and Vitali, the legal expert must meticulously and manually conduct the process – sorting within legal documents the text, structure, metadata, and ontology. As well, the legal expert must be fluent in Akoma-Ntoso, correctly annotating the elements and identifying the legal relationships latent in the documents.

[713] "Use Cases," Akoma Ntoso, available at: http://www.akomantoso.org/?page_id=275.

[714] Palmirani and Vitali describe in further detail the process of taking text and structuring. *See* Palmirani and Vitali, *supra* 698 at 94-98.

In effect, though Akoma-Ntoso offers benefits of making legal language machine-readable and preserves the richness of legal concepts, its use requires significant costs. The process is rather laborious, and few legal experts[715] currently have the technical skills to draft in XML schema. Consequently, this has contributed to rather lackluster enthusiasm for its adoption.

D. Old Wine in New Bottles: Rules as Code

Still, machine-readable legislation has received renewed popularity. This is perhaps owed to the release of the recent OECD Observatory of Public Sector Innovation Report titled, "Cracking the Code: Rulemaking for Humans and Machines" (OECD Report). The OECD Report articulates how machine-consumable, defined as machines understanding and actioning rules consistently, reduces the need for individual interpretation and translation[716] and "helps ensure the implementation better matches the original intent."[717] This methodology enables the government to produce logic expressed as a conceptual model – in effect, a blueprint of the legislation.

These ideas are reminiscent of Anthony Casey and Anthony Niblett's thought experiment on the micro-directive.[718] Interestingly, one of the underlying fascinations with Rules as Code lies in the types of statutes subject to digital transformation. Rules as Code applies two general practices of code-ification: (1) programming tasks; and (2) knowledge-based systems. The former is more direct, while the latter poses epistemic challenges. Programming tasks may be defined as a legislative calculator; the legal questions asked are already known and understood in advance. Typically, these tools are designed to assess eligibility, particularly in the fields of taxation and benefits law. OpenFisca, the most widely known example, is an open-source platform that writes rules as code. The available code focuses on legislation that "can be expressed as an arithmetic operation."[719]

---

[715] It must be noted that the XML vocabulary and schemas are open-source and publicly available. This suggests that while the documentation is available, it continues to remain limited amongst those willing to adopt the practice. See for example, OASIS LegalDocumentML, *supra* 693.

[716] OECD Observatory of Public Sector Innovation, *Cracking the Code: Rulemaking for Humans and Machines* 19 (2020).

[717] *Id.* at 22.

[718] To recall, in this futuristic construct, lawmakers would only be required to set general policy objectives. Machines would bear the responsibility to examine its application in all possible contexts, creating a depository of legal rules that best achieve such an objective. The legal rules generated would then be converted into micro-directives that subsequently regulate how actors should comply with the law. Anthony J. Casey & Anthony Niblett, *The Death of Rules and Standards*, Coase-Sandor Working Paper Series in Law and Economics No. 738 (2015).

[719] "Before You Start", Open Fisca Documentation (accessed January 2021) https://openfisca.org/doc/. For further details on how to 'translate' from law to code, *see*: https://openfisca.org/doc/coding-the-legislation/index.html.

Knowledge-based systems, on the other hand, encode rules required to arrive at a specific legal question. That is, these tools consist of logical algorithms that help identify the legal knowledge to be gathered from a particular statute. They come from the lineage of expert systems and logic programming. DataLex Knowledge-Base Development Tools (DataLex), for instance, is a rules-based legal inferencing platform that draws, from legislative texts, conclusions based on antecedents. In effect, the DataLex software is powered on propositional logic.[720]

Despite differences between practices of code-ification, the types of legislation amenable to a Rules as Code approach predicate on an inherently mathematical structure. This suggests that for legislation with clear formulaic rules, expression in symbolic logic is intrinsically available. Ruleness becomes the essential ingredient. The OECD Report, however, does not distinguish between types of legislation and, rather, conflates legislation under a seemingly uniform banner.

Though the OECD Report succeeded in providing a comprehensive overview of Rules as Code, there remains a gap around the practical implementation and the form machine-readable legislation should take. The OECD Report anticipates three approaches to building machine-consumable legislation: (1) a manual coding of the legislation across a multidisciplinary team; (2) the use of semantic technologies; and (3) a domain model-based regulation, whereby the government would create an official model of rules to then convert to software languages.[721] These approaches drew inspiration from a deeper analysis on the levels of digitization.[722] Unlike Meng Weng Wong's aspirational vision for machine-readability, the OECD Report is agnostic to these possible methods.

Recent implementations of Rules as Code have surfaced globally. Currently, the most prominent example is found in Australia. In the summer of 2020, the New South Wales (NSW) Government released its first Rules as Code legislation to reduce ambiguity and simplify interpretation.[723] Built on the OpenFisca platform,[724] the *Community Gaming Regulation 2020* (Gaming Regulation) identifies

[720] "Legal Inferencing Systems: Supporting provision of free legal advisory services," DataLex (accessed January 2021) http://austlii.community/foswiki/pub/DataLex/WebHome/DataLex_intro.pdf.

[721] OECD Observatory of Public Sector Innovation, *supra* 716 at 63-66.

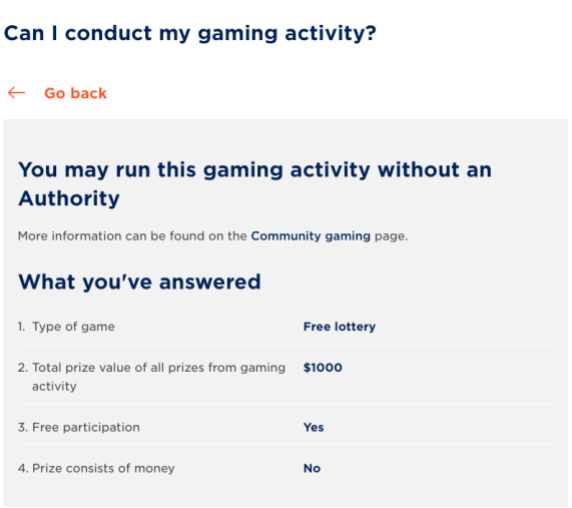
[722] Meng Weng Wong, *Rules as Code – Seven Levels of Digitisation*, Research Collection School of Law (2020).

[723] "In an Australian first, NSW is translating rules as code to make compliance easy," NSW government digital.nsw (accessed Jun. 12, 2021), https://www.digital.nsw.gov.au/success-stories/australian-first-nsw-translating-rules-code-make-compliance-easy.

[724] To see the regulation housed on the OpenFisca platform, *see* Openfisca-Nsw-Base Web API (accessed Jun. 12, 2021) http://nsw-rules-dev.herokuapp.com/swagger.

"the conditions for running community games by different charities, not-for-profits and businesses in NSW."[725] The *Gaming Regulation* is drafted in several forms: machine-readable, human readable, and on a computing interface. Perhaps its most incredible achievement is the publicly available digital version of the *Gaming Regulation*. The NSW Fair Trading website enables those engaging with the regulation to determine whether their activity is permissible and if an authority is required to conduct the activity.[726] This website is considered a "single source of truth" that will increase transparency and efficiency, by reducing time spent understanding the regulation, and providing easily digestible responses to particular situations of concern.[727] The website offers information on various sections of the legislation in plain language. The prize jewel, however, is its questionnaire.

In experimenting with the website's questionnaire, the "Community Gaming Check,"[728] the key content behind the legislation appears to be logically reducible and fundamentally arithmetic. Below are two sample snapshots of completed questionnaires:



Presumably, for the purposes of simplification, the questions are either drafted in binary or are numerically driven. As a result, the Community Game Check (CGC) will compute a response in the affirmative or negative. The underlying assumption of the CGC is that the legislation raises one of

[725] *Id.*

[726] *Id.*

[727] *Id.*

[728] For further detail and/or to experiment with the questionnaire, *see* "Community gaming check," NSW Government Fair Trading (accessed Jun. 12, 2021), https://www.fairtrading.nsw.gov.au/community-gaming/community-gaming-regulation-check. For the machine-readable version of the legislation, *see* Openfisca-NSW (accessed May 10, 2021) https://github.com/Openfisca-NSW/openfisca_nsw_community_gaming.

two questions: (1) determining whether a community game is admissible; or (2) if authority is required. Again, it may be reaffirmed that Rules as Code focuses on prescription and rules; description continues to fall within the jurisdiction of the original natural language version. Underlying this focus is the assumption that legislation is largely mathematical and that legislative questions may be solved through predicate logic.

Alternatively, the Rules as Code initiative sparked more granular innovations, including formal languages compatible for its drafting and expression. Catala, "a new programming language created by lawyers and computer scientists for quantitative statute formalization,"[729] is a proposed solution for computing tax and benefits legislation. In their article, Denis Merigoux and Liane Huttner explore the issues of existing expert systems used for tax and benefits law. They first outline that the use of antiquated code – programming languages that "exceeded the tenure of its original programmers"[730] – risks the inability of adapting to new functional demands. This has evident ramifications provided the evolving nature of legislation. Equally, they explore the pitfalls of using existing algorithmic tools for tax collection that has led to both miscalculations and barriers with revision.[731]

Their recommendation is to use formal methods coupled with literate pair programming in order to tackle the aforementioned issues. First, literate pair programming is a hybridized understanding of literate and pair programming in software development.[732] Merigoux and Huttner suggest that a combination of these methods, and between a lawyer and computer scientist, enable quality assurance in the translation of law to code. The line-by-line annotation of statutory texts allows for a "local discussion" on the "lawful interpretations of the statutes."[733] Evidently, this recommendation aligns closely with one of the OECD Report's anticipated approaches to building machine-consumable legislation: a manual coding of the legislation across a multidisciplinary team. However, the more pressing question is the use of formal methods.

---

[729] Denis Merigoux and Liane Huttner, *Catala: Moving Towards the Future of Legal Expert Systems*, HAL ARCHIVES-OUVERTES (2020).

[730] *Id.* at 2.

[731] *Id.* at 3.

[732] Literate programming is described as line-by-line annotations, while pair programming is pairing two programmers in the production of code. For further detail, see *id.* at 7.

[733] *Id.*

Formal methods are a restructuring of abstract concepts to "mathematical objects."[734] Formal methods act as mathematical proofs, determining functional equivalence.[735] Effectively, it is reminiscent of Carnap's logical syntax and treatment of language as a calculus. As a result, this practice depends on the existing and inherent formal structure of the legislation.[736] This again reinforces the requirement of ruleness in Rules as Code. Consequently, while Merigoux and Huttner's recommendations ensure that legal quality is maintained, Catala's benefits remain within the limited scope of intrinsically quantifiable legislation.

E. Legislative Tinkering

These recent implementations of Rules as Code fortify the argument that, currently, machine-consumable legislation is limited to highly structured legislation. Nevertheless, these examples leave one question fundamentally unanswered: *what should be the role of machine-readable legislation?* Is it simply a 'coded' version of the legislation; or is it a parallel alternative, one that is legally authoritative? Or is it a domain model of regulation from which third parties derive their own versions, akin to an open-source code? These three scenarios have their own sets of implications. Only in clarifying the role of machine-readable legislation would a fruitful assessment of how logic syntax and symbolic language are capable of representing legal knowledge.

i. Authoritative Conundrum

New Zealand released in March 2021 its own version of the OECD Report, "Legislation as Code for New Zealand: Opportunities, Risks, and Recommendations" (Legislation as Code Report). One of the key conclusions of the report calls for a distinction between competence and desire. That is, even if legislation may be drafted in code, it should not be. Unlike the OECD Report, the Legislation as Code Report takes a strong stance on the role of machine-consumable legislation. The report argues that rules drafted in code "should remain subordinate to legislation," stating that "enacting code creates serious constitutional confusion and risks undermining the separation of powers."[737]

---

[734] *Id.* at 6.

[735] *Id.*

[736] Merigoux and Huttner state explicitly the assumption of expression in mathematical terms as well as the "formal specification" of statutes. See *id.*

[737] New Zealand Law Foundation Law and Information Policy Project, *Legislation as Code for New Zealand: Opportunities, Risks, and Recommendations* 3 (2021).

This is owed to the law's "technological use of written natural language;" whereby the use and interpretation of words keeps in balance the structure of the law with its institutions.[738] As code does not have the same interpretive space as natural language, this runs the risk of the judiciary being unable to perform its constitutional role relative to statutory interpretation.[739] Accordingly, the inability to invalidate legislation for inconsistency, given interpretative barriers with code, would "degrade the rule of law."[740]

ii. Language Shopping

The Legislation as Code Report further contrasts the OECD Report by concluding that parallel drafting is not a solution, but simply a mitigator to issues of interpretation.[741] Provided that perfect translation does not exist, there is inevitably potential for meaning to diverge even if a common intent is established. Therefore, while an encoded version arguably reflects *an* interpretation of the law,[742] machine-consumable legislation that has legal authority raises, equally, issues analogous to both legislative bilingualism and bijuralism.[743] This could foreseeably create statutes with multiple personalities, having dissonance between linguistic variants and heightening ambiguity in interpretation.

In this regard, Canada is an informative example. In 1995, the formal adoption of legislative bijuralism led to an acknowledgment of four legal audiences in Canada; that there is a "right to read federal legislation in the official language of their choice and to find that legislation terminology and wording [to be] consistent with the system of private law in effect in their province or territory."[744] As such, the constitutional requirement for all legislation to be written bilingually forcibly produced makeshift equivalents in legislation, devised without standard nor appropriate concern for the problems of interpretation.

---

[738] *Id.* at 9.

[739] *Id.* at 58.

[740] *Id.*

[741] *Id.* at 4.

[742] In fact, the Legislation as Code Report suggests that it may be useful to focus on the opportunities for approaches of non-authoritative implementations of Rules as Code. See *id.* at 9.

[743] Lionel A. Levert, "Harmonization and Dissonance: Language and Law in Canada and Europe," Department of Justice Canada, *Bijuralism and Harmonization: Genesis* (May 7, 1999) https://www.justice.gc.ca/eng/rp-pr/csj-sjc/harmonization/hfl-hlf/b1-f1/bf1e.html.

[744] *Id.*

There are two models of producing bilingual legislation: translation and co-drafting. While they are perceived as distinct, the process around crafting bilingual legislation often involves a hybridization of both. This typically results in a conceptual mismatch between one language to the other. Michael J. B. Wood provides a fascinating illustration through the word 'any.'[745]

(1) The report shall include **any** document specified in the schedule.	(1) La rapport comprend **l'un des** documents énoncés à l'annexe. (1) Le rapport comprend **les** documents énoncés à l'annexe.

In the English language, 'any,' in the affirmative, describes 'one' out of a specific list. In the above example, the intention of the drafter may be to indicate that, should there be documents specified in the schedule, they should be included. However, to the reader, it may suggest that any one of the documents specified in the schedule should be included. Consequently, in the French language example, there produces two variants. This lack of equivalence in the word 'any' produces ambiguity between versions of the legislation. Both of which have equal authority under Canadian law. Wood discusses other examples including pronominal phrases such as 'thereof' and chains of qualifiers.[746] In the former, phrases of this type often foster confusion, particularly in co-referencing.[747] As well, there are no direct equivalents in French. In the latter, the Germanic origins of the English language allow nouns and adjectives to be chained together. This use of grammar does not exist in French. Instead, the French language applies a series of modifying phrases. Consequently, if meaning is unclear and ambiguous in English, there is potential for further complication in French.[748]

Likewise, the presence of both civil and common law systems within Canada has led to complications with the translatability of legal concepts. Bijuralism stipulates the requirement to have proper terminology and notions present across both systems of private law in Canada. To achieve this requirement, the most frequent methods used are the "neutrality technique" and the "doublet."[749] The former is simply the use of 'neutral' terms or phrases in defining concepts without particular

[745] Michael J.B. Wood, *Drafting Bilingual Legislation in Canada: Examples of Beneficial Cross-Pollination between Two Language Versions*, 17 Statute. L. Rev 66, 70 (1996).

[746] *Id.* at 70-72.

[747] *See* example on "a part thereof." *Id.* at 70.

[748] *Id.* at 71-72.

[749] Levert, *supra* 743.

connection to either one of the systems. The latter is to enable the co-existence of legal concepts when there is no functional equivalence. In cases of the doublet, both versions of the legislation "retain their separate identities."[750] This means that paragraphs within the same legislation may have intentional signposts to direct how the rule of law is to be applied depending on the system.[751] Typically, both expressions of the legal concept appear one after the other in each language version.

Evidently, problems of interpretation arise as "civil law terms are juxtaposed with common law expressions."[752] Within the country, there were issues symptomatic of conflict of laws; whereby courts applied common law definitions to jurisdictions that followed civil law systems. This led to inconsistencies in precedent, as civil and common law terminology were used interchangeably without proper regard for the nuances of legality between each system's interpretations.

Canada has since made remarkable strides in legislative bilingualism and bijuralism. This was owed to a reframing of federal requirements as a strain of comparative law, as well as the subsequent emergence of jurilinguists; otherwise, experts trained in both systems.[753] Returning to machine-readability and authoritative code, what are some lessons that can be drawn from the Canadian experience? First, there has been a rise in interdisciplinary training between law and computer science. Mireille Hildebrandt's recent textbook is a prime example. *Law for Computer Scientists and Other Folk*, as she describes, is an endeavor to "bridge the disciplinary gaps" and "present a reasonably coherent picture of the vocabulary and grammar of modern positivist law."[754] As well, law schools are beginning to offer technology and innovation courses including training in computer

[750] *Id.*

[751] *Id.*

[752] *Id.*

[753] Universities of Ottawa and departments of jurilinguistics produced both common law terminology in French and civil law terminology in English. This pioneering work offered the potential to better capture the necessary distinctions and comparisons between the two systems of law. See *id.*

[754] Mireille Hildebrandt, *Law for Computer Scientists and Other Folk* (forthcoming OUP, 2020). A web version is currently accessible on the open-source platform: https://lawforcomputerscientists.pubpub.org/ .

programming.[755] This is facilitating a growth and demand in experts fluent in both disciplines.[756] Moreover, as evidenced, co-drafting can be seen in the recommendations and development of machine-readable languages like Catala.

There remains, however, a significant gap in both reconciling and harmonizing legal concepts between code and natural language. Perhaps the deeper question is whether and how that may be possible. In Canada, common and civil law terminology come from existing traditions of private law. Their respective expressions are rooted in legal history. However, there is neither a comparable legal system nor a comparative field of law for code. That is, code could only potentially extend as an alternative language, but not as a system of norms. The functional limitations of code could only be interpreted as linguistic limits, whereas normative principles of programming and computer science could never be perceived as parallel legal principles. As a result, the discussion raised in the Legislation as Code Report, on the risk of authoritative code degrading the rule of law, is a critique of code as a legal mechanism. The complexity lies in the extent to which the linguistic medium has the capacity to alter the integrity and character of the law, even if the intention of its use is simply expression.

iii. The Alchemy of Legal Architecture

Perhaps the most understated challenge with Rules as Code hinges on the legal infrastructure. Across several possible approaches to machine-readable legislation, there remains unresolved questions of design and interoperability between legal documents. That is, if a new symbolic language, like code, effectively enforces a controlled grammar, what are its implications as it moves across the legal ecosystem; in particular, its interactions with various legal sources?

Reflecting back on the Legislation as Code Report, one important argument raised is the acknowledgment of legislation as "one component among many that comprise the wider system of

[755] Law schools are beginning to offer courses in technical development, including computer programming. Moreover, classes that apply design-thinking to legal studies and were developed with the intention of acknowledging technology as a powerful driving force in law. Consider Harvard Law School and Georgetown Law School's Computer Programming for Lawyers classes, or Innovation Labs at Northwestern Law School or The Design Lab at Stanford Law School. See for example Harvard Law School, *Computer Programming for Lawyers* (accessed February 2020), https://hls.harvard.edu/academics/curriculum/catalog/default.aspx?o=75487.

[756] "Embedded technical expertise may be necessary to design, develop, and maintain useful and useable tools," also "development of the tool resulted from a multi-year strategic plan to hire lawyers with coding skills..." *See* David Freeman Engstrom and Daniel E. Ho, "Artificially Intelligent Government: A Review and Agenda" in Roland Vogl (ed.), *Big Data Law* (2020).

laws and rules."[757] Statutes frequently reference one another, highlighting a "process of synthesizing multiple inputs into a contextually dependent output."[758] Provided that legislation are not perceivably independent texts, it is then important to consider how machine-readable legislation works in tandem with other legal documents.

In the OECD Report, the discussed approach for a domain model-based regulation is one that raises persistent queries on interoperability. Should there be a government-endorsed model from which legislation will be converted into third-party machine-readable versions, this could create inconsistent interpretations; thereby, testing the legal limits of the model. Currently, there is no standard for how the model translates to individual policies. More importantly, what might be issues of fit between various machine-readable documents, such as between machine-readable legislation to machine-readable contracts?

In late December 2020, the University of Cambridge announced the launch of the Regulatory Genome Project.[759] As opposed to legislation, the focus of the project is on regulation, and specifically financial regulation. The Regulatory Genome Project intentionally steers away from regulation as code and considers the notion of "sequencing."[760] Rather than translation, regulatory information will be extracted and placed in a data repository. The regulatory data will then be organized into a taxonomy. In accordance with the taxonomy, experts will annotate key information and build a training set. This model will then be used to subsequently generate machine-readable regulatory documents. In effect, it is a process of retrieving the contents of regulation from an openly accessible platform that bears a specific framework of capturing the regulatory data. This permits a single source of 'truth' and a common standard for accessing machine-readable regulatory information.

The significance of this approach is its departure from language design. That is, as opposed to dwelling on the semantic conversion of natural language to code, the project turns its attention to the information contained in regulation. It is simply a complete rewrite, or paradigm shift, of digesting regulation. Beyond an interdisciplinary collaboration, the Regulatory Genome Project has received

[757] Legislation as Code Report, *supra* 737 at 48

[758] *Id.*, at 50.

[759] "The University of Cambridge announces the launch of the Regulatory Genome Project to sequence the world's regulatory text through machine learning," *The Regulatory Genome*, https://regulatorygenome.com/news/university-cambridge-regulatory-genome-project/.

[760] The Regulatory Genome Project, *The Regulatory Genome* (accessed March 10, 2021) https://regulatorygenome.com/about-us/.

the support of regulators, authoritative figures of the community, "to validate and refine the taxonomies to enable effective benchmarking across jurisdictions globally."[761] Interestingly, this parallels an amalgam of the Rules as Code domain-model with the Legislation as Code argument that the variability of interpretations would be limited if authoritative interpretations are made available.[762]

As a result, the Regulatory Genome Project offers an unconventional method for machine-readability. Evidently, this may be simpler with regulation than it is with legislation. Namely, legal authority operates differently than regulatory authority. In considering this approach, the challenge would be systemic and one that requires convincing a complex network of legislative and judicial power to construct laws on an entirely separate paradigm. Nonetheless, it offers a perspective on mediums of communication and computational modelling that extends beyond language to a level of further granularity: data.

Existing literature has focused on the promise of Rules as Code as the magical formula for increased clarity and precision in legislative drafting. Undeniably, machine-readable legislation has deep-seated roots in logical syntax and symbolic language. The Legislation as Code Report, however, highlights that further discussion is required in better defining both the legal function and status of machine-consumable legislation. Fundamentally, machine-readable legislation requires a space for judicial and legal contest; effectively, an appeal process in the event of dispute.[763]

This is not to say there is no place for machine-readable legislation. In fact, the Legislation as Code Report argues that computational models can be commendable if the model is (1) "legally correct," and (2) there is infrastructure in place "to assess how the law has been interpreted and modelled."[764] For example, the Legislation as Code Report cites the Auckland District Law Society's Standard Form Agreement for Sale and Purchase of Real Estate (ADLS Standard Form). The ADLS Standard Form is described as an instrument that "embod[ies] a reliable interpretation of multiple primary legal sources" and "indicate[s] the value that similar interpretation might have if they are coded and

[761] *Id.*

[762] Legislation as Code Report, *supra* 737 at 82.

[763] This is reminiscent of the argument raised in Kiel Brennan-Marquez and Stephen Henderson's article on concept of role-reversibility integral to the legal system. *See* Kiel Brennan-Marquez and Stephen Henderson, *Artificial Intelligence and Role-Reversible Judgment*, 109 J. CRIM. L. & CRIMINOLOGY 137 (2019).

[764] Legislation as Code Report, *supra* 737 at 5.

modelled reliably, while retaining the ability to scrutinize them through legal argument."[765] Provided that this agreement has been drafted and revised within a dependable legal environment, the ADLS Standard Form has demonstrated the potential for reproducibility while maintaining certainty. This suggests that finding existing natural language documents with an accepted standard and structure may be appropriate for computational modelling.[766] Again, this reinforces that Rules as Code is available only in narrow-use cases, specifically, legislation with inherent logical structures.

At a broader epistemological level, there remains limitations from the perspective of knowledge representation; in turn, forcibly demanding a reflection on the intentions and purpose of laws. The Regulatory Genome Project has revealed that there may be an alternate option of consuming information. As law has language at its core, interpretation has centered on the linguistic exercise. This has led to a heavy reliance on translation when reconciling human with machine-readability. However, lessons from core linguistics suggest that natural language is composed of three underlying components: syntax, semantics, and pragmatics. Curiously, the enduring focus on the syntax and semantics in computational models has led to a subsequent neglect of pragmatics, an arguably essential pillar in meaning-making. Consequently, this impedes the capacity to appropriately understand and contextualize legal concepts.

To recall, pragmatics is the field of linguistics that reflects on intention using tools of implicature and inference. Implicature, in linguistics, is defined as entailment, logically valid conclusions drawn between sentences.[767] Its counterpart, inference, is more complex. This is where discrepancies may exist, as what is being implied may differ from what is inferred. In accordance with Grice's Cooperative Principle,[768] the divergence between intended implicature and inference suggests non-conventional meaning. In effect, this supports the possibility of multiple interpretations on the basis of variations in context.

Consider the phrase: "There is an elephant in the tree." Semantics is helpful, to the extent, that it could raise what may be a prototype example of an elephant. As elephants are not typically found in

[765] *Id.* at 83.

[766] The discussion by Sarah Lawsky furthers the support for form as a ripe area of formalization. *See* Sarah Lawsky, *Form as Formalization*, Ohio State Tech. L. J. (forthcoming 2020), available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3587576.

[767] Betty J. Birner, *Language and Meaning* 102 (2018).

[768] *Id.* at 96.

trees, this is immediately a sign that this sentence may have a different meaning. Could this be a metaphorical idiom (i.e. elephant in the room) or perhaps there is some implicit understanding that the elephant in question is a paper elephant? Pragmatics also raises the issue of reference. Consider the following sentences: "Jane is speaking with Joanne. She is a legal scholar."[769] The referent of "she" is not clear. Without context, semantics alone cannot usefully provide information as to the meaning of these sentences.

There are parallels to the shortcomings of semantics revealed in propositional logic. Systems that use propositional logic, similar to Rules as Code structures, reflect the limitations presented in semantics. This is because propositional logic can enable the validation of some statements but cannot in itself establish the truth of all statements. So, why must there be consideration for pragmatics in machine-readable legislation?

Joseph A. Grundfest and A.C. Pritchard discuss the "technology of ambiguity" as a legislative strategy for compromise.[770] Their article reaffirms the notion of intentional, conscious, ambiguity. As opposed to ambiguity as a 'bug,' Grundfest and Pritchard argue that it is feature of legislative drafting. That is, ambiguity in the drafting process is intended to work in tandem with the judiciary's interpretative methods. Ambiguity then works to ensure that the casuistic approach, characteristic of common law systems, is upheld.

Contrary to the rhetoric on clarity and precision, ambiguity is revered as an inherent property of statutory construction. While this is not necessarily a novel argument, Grundfest and Pritchard reassert the interoperability of the legal system; legal documents are not independent artifacts and instead belong to a broader ecosystem. The aforementioned issues of pragmatics in natural language are integrated into the fabric of law and legal text and powered by literary tools of metaphor and analogy that outline context.

Interestingly, code is not quite as transparent or reducible as assumed. Mark C. Marino argues that code, like other systems of signification, cannot be removed from context. Code is not the result of mathematical certainty but "of collected cultural knowledge and convention (cultures of code and coding languages), haste and insight, inspirations and observations, evolutions and adaptations,

[769] Drawn from Birner's example. *Id.* at 109.

[770] Joseph A. Grundfest and A.C. Pritchard, *Statutes with Multiple Personality Disorders: The Value of Ambiguity in Statutory Design and Interpretation*, 54 STAN. L. REV. 627 (2002).

rhetoric and reasons, paradigms of language, breakthroughs in approach, and failures to conceptualize."[771] While code appears to be 'solving' the woes of imprecision and lack of clarity in legal drafting, the use of code is, in fact, capturing meaning from a different paradigm. Rather, code is "frequently recontextualized" and meaning is "contingent upon and subject to the rhetorical triad of the speaker, audience (both human and machine), and message."[772] It follows that code is not a context-independent form of writing. The questions become whether there could be a pragmatics of code, and if so, how could code effectively communicate legal concepts?

Marino articulates the "need to learn to read code critically."[773] Having understood the complexities and pitfalls of natural language, there is now a rising demand to understand the ways code acquires meaning and how shifting contexts shape and reshape this meaning. Currently, few scholars have addressed code beyond its operative capacity. This mirrors the focus on syntax and semantics as primary drivers of using code for legal drafting. Yet, learning how meaning is signified in code enables a deeper analysis of how the relationships, contexts, and requirements of law may be rightfully represented. From the science of (natural) language arises the science of code.

Increasingly, there has been emerging literature on the application of network analysis and graph theory to account for legal complexity. In a recent article on the growth of the law, representations of legislative materials were modelled using methods from network science and natural language processing.[774] Katz et. al argue that quantifying law in a static manner fails to represent the diverse relationships and the interconnectivity of rules. They suggest that statutory materials should instead be represented using multidimensional, time-evolving document networks. As legal documents are interlinked, networks better reflect the dynamics of their language and the "deliberate design decisions made."[775] Moreover, it enables "circumvent[ing] some of the ambiguity problems that natural language-based approaches inherently face."[776] Most fascinating is the authors' capacity to isolate, through graph clustering techniques, legal topics that have fostered the most "complex bodies

[771] Mark C. Marino, *Critical Code Studies* 8 (2020).

[772] *Id.* at 4.

[773] *Id.* at 5.

[774] Daniel Martin Katz et. al., *Complex societies and the growth of the law*, Sci Rep 10, 18737 (2020), available at: https://doi.org/10.1038/s41598-020-73623-x.

[775] *Id.*

[776] *Id.*

of legal rules."[777] This enabled a deeper understanding of the evolution of legal concepts and specific points of inflection where their perceptions have shifted.[778]

What is particularly striking about this paper is the introduction of quantitative approaches that stress content representation as opposed to structural miming. This model considers importantly context that shapes legal documents. How then could machine-readability be reconciled with graphical representation of legal documents? Statutory and legislative materials necessarily are situated at the heart of the legal ecosystem. That is, legislative documents provide the foundation on which other legal documents could gather concepts. This suggests that as opposed to an emphasis on semantic translation to machine-readable legislation, a consideration of the role of legislation from an information extraction perspective may be a promising alternative.

CONCLUDING REMARKS

In analyzing the 'coming-of-age' of machine-readability, it becomes striking clear that, even with current advancements, there remains a gap around its role vis-à-vis 'human-readable' legislation. The complexity of translating legislation from natural language to code stems from a persistent conceptualization of legal documents as independent entities. Rather, legal information must be understood at a systemic level; to factor the interaction of legal documents with one another across a temporally sensitive frame. Therefore, legal texts should be perceived as objects with code as the semiotic vessel. How these objects interact, how references are made, and how their histories interrelate must be accounted. It appears then that a dual-pronged method of semiotic analysis coupled with pragmatics contribute to a more fruitful engagement of legal knowledge representation. As opposed to applying an arithmetic lens in the name of clarity and precision, language design for machine-readability requires a multi-layered approach that extends beyond syntactic structure and ensures temporal management and formal ontological reference. Without these considerations, machine-readable legislation could only remain in the realm of a computable iteration.

In the remainder of the thesis, I reconcile prior literature and thematic discussions with observations from the case studies. It is in these chapters that I consider the future of computational law. I do so

[777] *Id.*

[778] Consider the discussion by the authors on the regulation of natural resources from exploitation to conservation. See *id.*

by clarifying whether natural language is indeed the only linguistic medium for legal conveyance, or, whether we may be at the frontiers of a new linguistic medium.

4- Weaving the Code

Recall in *The Linguistic Affair* the discussion on the notions of conceptual transfer and intersubjectivity. That is, can concepts be transported and migrated from one vehicle to another? Evidently, the deconstructionist perspective suggests that this is not possible. Nevertheless, the various case studies demonstrated that, in certain respects, a hybrid or layering approach may be an opportunity as an intermediary (or, transitory) step. Simply put, certain tasks may be code-ified, while others must continue to rely on natural language construction. The process will be one of sorting and authoritative assessment.

Alternatively, the advent of computational contracts and machine-readability has accelerated the pressure for a new form of legal expression, particularly one of heightened precision and accuracy. While I believe that natural language would continue to be the dominant form of legal conveyance, this section attempts to put forward a working hypothesis around reconciling code as the next legal language.

The second case study had attempted to experiment with the deconstruction of legal text; how breaking down natural language into its core components fosters translation into code. Notably, this was an immense interdisciplinary effort. It required the joining of several disciplines, including mathematics, data science, and linguistics to carefully unpack the complexity of judicial texts. The result, of course, had led to fascinating discoveries around the jurisprudential patterns and mechanisms of legal language. Nevertheless, it revealed two key elements: (1) linguistic fingerprints; and (2) a multi-computational strategy. The former points to the syntactic and semantic markers that provide the building blocks around legal grammar. More importantly, it reinforces the indispensable need for linguistic analysis in legal writing. The latter alludes to the misconception of computation as one-dimensional, instead highlighting that the complexity of language necessarily requires more than one computational tool in the work of translation.

As opposed to a 1-to-1 mapping, or broad analogies[779] around computation and law, the relationship between law and technology is far more nuanced. That is, for the furtherance of computational law, there must be a more granular practice in place. Consequently, for law to be expressible in a computable form, there requires a better representation of pragmatics. Currently, programming

[779] Predictive analytics is similar to analogical reasoning. Expert systems are like syllogisms. Natural language processing is like contract redlining.

languages fail to include context, and in effect, are unable to infer beyond sentential understanding.[780] This is incredibly problematic as the legal language is notably riddled with reference beyond the text. Moreover, the inability to account for pragmatics equally reflects the incapacity to apply figurative and metaphorical language. This results in current computable forms of "law" that are reduced to logic and structure. This fosters an incongruency and enables "bad translations" of legal text to code.

Programming languages are built on syntax and semantics. While there are evident differences between syntax and semantics in core linguistics and programming, both predicate on context independence, logic, and universality. In effect, this has led to reformulations of legal norms as "objective" truths. The problem is that the law is built on both facts and norms. Setting aside the added complexity of law's fictional character, prioritizing syntax and semantics dangerously asserts that all law is fact. As a result, the law shapeshifts away from a bidirectional relationship between framing and restoring order to a unidirectional relationship of compliance. Therefore, it is my assertion that legal concepts have been housed well in natural language because of the significant role played by pragmatics.

To then attempt an exercise of conceptual transfer, and appropriately reflect on the limits of legal expression, there must necessarily be consideration for how pragmatics may be reflected computationally. Translation and the authoring of legal text must first evaluate how (1) inference and embedded knowledge revealed in natural language can be modelled; and (2) how code as a non-natural and non-linguistic vehicle conveys context. I will rely on the texts, *The Myth of Artificial Intelligence* by Erik J. Larson and *Critical Code Studies* by Mark Marino as references. The former will assist with debunking the puzzle of pragmatics and inference, and the latter for introducing a semiotic understanding of code and programming.

The remainder of the chapter will proceed as follows. First, observations from the case studies will be discussed in further detail. Reflections on congruencies between human- and machine-readable text, their respective assumptions, and current treatment will be highlighted. Considerations for the future of contracts and legislative drafting, as well as the persistence, and perhaps resilience, of

[780] Consider for example literature from Emily M. Bender on the distinction between form and meaning. Specifically, the capacity to map structural patterns should be distinguished from the ability to understand. This has parallels with my observations on the existing arguments around form and substance in law and its language. *See* Emily M. Bender and Alexander Koller, "Climbing Towards NLU: On Meaning, Form, and Understanding in the Age of Data," *Proceedings of the 58th Annual Meeting of the Association of Computational Linguistics* (July 2020) available at: https://aclanthology.org/2020.acl-main.463/.

natural language will also be analyzed. Next, the chapter will introduce the problem with inference, then proceed with a thought experiment on code as the new medium of legal language. In other words, how can we formulate a pragmatics of code? While I certainly cannot and do not intend to claim that this could be the working model, I nevertheless seek to draw attention beyond the gaps and towards potential methods of developing a legal semiotics.

Faux Amis and Hybrid Forms

In learning French as a second language, native English speakers are quickly alerted to the risks of *faux amis*. "Faux amis," or false cognates, describe words that look similar in both languages, but, in fact, have different meanings. For example, the *attendre* in French is not the same as *attend* in English. *Attendre* means to wait, while *attend* has multiple meanings, such as "to care for," "to deal with," or "to participate in." Likewise, the notion of machine-readability has introduced the issues of false cognates to legal drafting.

As presented in *Language Lego*, a pairing exercise has emerged whereby syntax and semantics in core and computational linguistics are treated as functionally interchangeable. Moreover, the implications have permeated across how computational technologies manifest in the legal realm. Consider the programming language, Lexon, from the first case study. In short, Lexon appears to draft contracts in a manner that is human-readable. That is, Lexon uses natural language constructions as their programming syntax. Their claim is that, just as in natural language, certain words are operative. In this case, the programming language is executable with their constrained grammar acting as triggers for contractual performance. However, Lexon "code and non-functional text are freely mixed."[781] This means that the programming language is syntactically significant and semantically void. Its 'readability' is derived from the surrounding contractual clauses and not the Lexon code itself. Divorcing the contractual "components" on the premise of utility reinforces the notion that code is task-oriented. As well, the functional/non-functional divide further implies that priority rests in the performance of the contract, reframing other language as 'noise.' This results in a conceptual rupture in contracts doctrine caused by forcibly 'translating' law to code.

This problem resurfaced in discussions on computable legislation, in particular Rules as Code. Rules as Code reinvigorated the enthusiasm around drafting legislation in code. The goal is to increase the

[781] "Lexon: Natural Language Programming," http://lexon.tech/ (accessed Jun 22, 2021).

transparency, clarity, and precision of legislative documents. The subtext, however, is that interpretative flexibility is a deficiency. That is, the fluidity of natural language has made it difficult to take stock of legal interpretation. Consider the example of Canada and the difficulty associated with interpreting legislation that is both bilingual and bijural. The incongruency of linguistic expressions, coupled with differing legal systems, have subsequently led to an internal conflict of laws. For example, Canadian courts have raised questions as to whether civil law concepts, drafted in the French language, are even translatable to English. Similarly, though Rules as Code purports to increase certainty, drafting in a programming language is akin to converting legal concepts simultaneously into a different language and system of norms. This appeared as a rather forthright exercise, provided that Rules as Code predicated on legislative documents that were inherently mathematical in structure. Consequently, this resulted in reframing legislative clauses to propositional calculus. Validity would indeed become synonymous with legality; in effect, closing the interpretative space through logical reduction.

Laurence Diver brings forth the concept of computational legalism, a digital twin to the tyrannical "acquiescence to rules as they are written."[782] Diver describes how computational legalism is fueled by both temporal and spatial decompression; a collapsing of the "hermeneutic gap" owed to the speed of code's execution.[783] He argues that the "ruleishness that is paradigmatic of code's character makes it immune to context."[784] Code, therefore, is an abolition of the normative space. In contrast, the delay enabled by text creates a gap for contestability and argument.[785] Text allows meaning to be indeterminate. The de-spatialization that Diver describes is effectively a regard of code as complete. It follows that code is perceivably incompatible with text, as they are artifacts of fundamentally conflicting systems. Therefore, as code cannot view legal concepts in the way that text (or natural language) can, translation is not possible.

However, Diver offers an alternative. He suggests that the use of code is possible to the extent that the architectural design compartmentalizes the technical and the human.[786] This is consistent with

[782] Laurence Diver, *Computational legalism and the affordance of delay in law*, J. OF CROSS-DISCIPLINARY RESEARCH IN COMPUTATIONAL LAW [CRCL] 6 (December 2020).

[783] *Id.*

[784] *Id.*

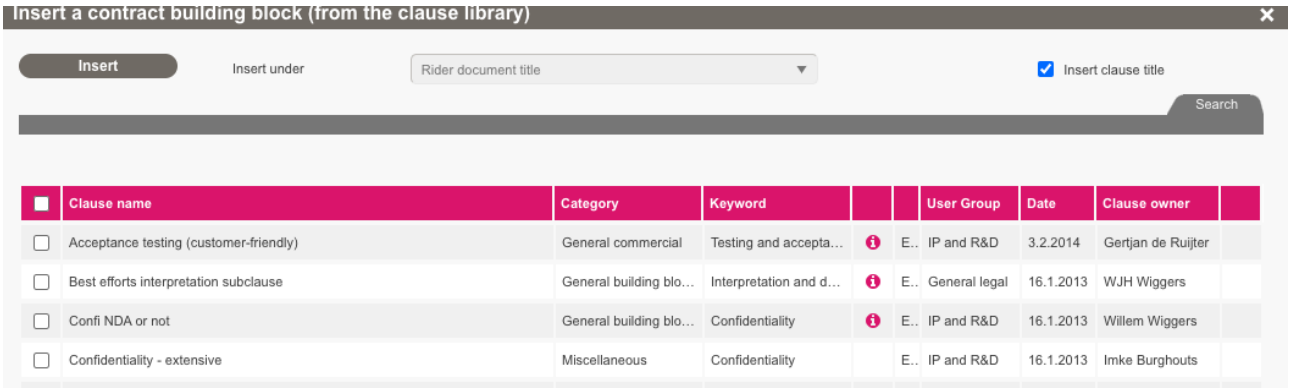
[785] Diver describes as "inviting dissent". See *id.* at 10.

[786] *Id.* at 9.

the notion of sorting or layering gathered from the case studies. With machine-readable legislation, LegalXML exemplified the opportunity to rearrange legislative documents into layers. Text is organized such that context and references are not lost. Instead, they are sorted into the 'metadata' layer. This enables interpretations of legislative clauses to be connected with their sources of legal authority, representing them as parts to the whole [legal ecosystem]. The challenge, of course, is the expertise required. This practice of 'layering' necessitates both legal and XML knowledge. As a result, with few experts that possess the skills required, current costs of LegalXML are rather significant.

Examples of sorting are also found in contract drafting technology. In the first case study, hybrid-programming languages, like OpenLaw, drew attention to the existing homogeneity in certain contractual clauses. Certain provisions are categorized as sufficiently standard (i.e., boilerplate) and with such little variance that, frequently, they are simply 'inserted' into the documents. OpenLaw uses genericism as a benchmark. The more generic the language, the more likely the clause may be code-ified. Unlike Lexon, embedding machine-readable code with natural language clauses is not necessarily translation. Instead, the code is perceived as an existing object that already belongs to the legal document. Rather than a rupture, there is conceptual continuity.

Startups like WeAgree have capitalized on hybrid forms by developing 'clause libraries:'[787]



WeAgree conceives of contractual clauses as reusable building blocks. Their idea is to foster party autonomy by extending outside of the document to the clause level. This allows 'boilerplate provisions' to be included in contracts that typically do not specify that type of clause (e.g., an

[787] Taken from WeAgree Wizard contract automation platform. *See* "Clause library integrated in contract automation," WeAgree: Accelerated Contract Flow, https://weagree.com/contract-automation/clause-library-integrated/ (Jun. 22, 2021).

intellectual property licensing clause in a confidentiality agreement). As a result, the treatment of code as an object maintains text at the forefront. This design choice prioritizes human centricity and maintains the integrity of the contractual process as a negotiated one.

What may be gathered is that the integration of code and text reflects an epistemological stance on legal interpretation. Fundamentally, machine-readability and the desire to translate text to code reinvigorates the notion that the law, in its current state, is uncertain and imperfect. The existence of the machine-readable variant then implies that code can resolve these defects. In short, law should be code. In contrast, hybrid forms consider machine-readable code as only secondary to natural language. Importantly, it suggests that, while code can offer benefits of efficiency, it does not regard efficiency as the goal. As a result, the layering approach then maintains the normative gap. As well, it circumvents problems associated with a code-driven law.[788]

Jeffrey M. Lipshaw considered the persistence of 'dumb' contracts,[789] or more simply, contracts drafted in natural language as opposed to code. Lipshaw clarifies that the intuition to restate contractual 'logic' into code is misleading. In his paper, Lipshaw experiments with translating Article 2 of the Uniform Commercial Code (UCC) to formal logic. Interestingly, he was able to formally prove that a buyer can be compensated for damages.[790] Moreover, Lipshaw notes that Article 2 includes fuzzy standards (e.g., "to sell goods that are fit for ordinary purpose").[791] Still, fuzzy logic was able to account for seemingly subjective criteria. This suggests that legal documents that involve complex future contingencies, albeit written in natural language, are already reducible to simpler more logical structures.[792] However, Lipshaw argues that imminency leads to risk-hedging behavior. In effect, vagueness, or 'elasticity,'[793] are pragmatic functions of natural language that create the strategic space for mitigation. Formal logic, on the other hand, is complete and unambiguous. There is no elasticity available.

---

[788] As discussed by Mireille Hildebrandt, "Code Driven Law Scaling the Past and Freezing the Future," Christopher Markou and Simon Deakin (eds.) in *Critical Perspectives in Law and Artificial Intelligence* (2020).

[789] Jeffrey M. Lipshaw, *The Persistence of "Dumb" Contracts*, 2 STAN. J. BLOCKCHAIN L. & POL'Y 1 (2019), available at: https://stanford-jblp.pubpub.org/pub/persistence-dumb-contracts/release/1.

[790] *Id.*

[791] UCC §2-315. See *id.*

[792] *Id.*

[793] Lipshaw cites linguist Grace Q. Zhang on strategic uses of elastic language. *See* Grace Q. Zhang, *Elastic Language: How and Why We Stretch Our Words* (2015). See also *id.*

Consider Relevance Theory[794] in linguistics. According to Relevance Theory, there are identifiably three levels of meaning: (1) logical form; (2) explicature; and (3) implicature. Meaning is derived from accessing all three levels. Below is an informative example:[795]

> "You are not going to die."
>
> *Logical form*: The receiver is immortal.
> *Explicature*: You are not going to die from this paper cut.
> *Implicature*: You are being dramatic and should stop making a fuss.

Notably, explicature and implicature are both pragmatic developments of the sentence's logical form. Explicature provides further detail that contextualizes the original sentence. This suggests that what is said cannot solely be derived from lexical meaning and syntactic combinations. Returning to Lipshaw, the assumption is that code, unlike natural language, is unable to 'enrich' propositions expressed, since formal logic has no pragmatic dimension. As a result, there will be a persistence of legal documents drafted in natural language. Though logic is evidently a core component to legal structure, logic lacks the elasticity that is currently only available in the natural language realm. More importantly, this perhaps justifies the compromise arrived at by the hybrid or layering approach. While logic is present, natural language text must persist to clarify meaning. Nevertheless, I will consider, further in the chapter, whether pragmatics can be represented computationally. For now, an unconventional paradigm will be explored to reflect on whether questions of natural language and code are, instead, an ontological problem.

A) Alternative paradigms: Semantic Interoperability and IEML

In 2020, Pierre Lévy introduced the Information Economy MetaLanguage (IEML). IEML is a computable semantics, capable of 'bridging' code with natural language. Lévy suggests that the incongruency between programming and natural language results from a lack of semantic interoperability. He states that while meaning is shared between languages, the expression of it differs.[796] Drawing from Chomsky's syntactic theory of Universal Grammar, Lévy imagines a

[794] *See* originally Dan Sperber and Deirdre Wilson, *Relevance: Communication and Cognition* (1986).

[795] Adapted from example in Carston. *See* for further detail, Robyn Carston and Seiji Uchida (eds.), *Relevance Theory* (1998).

[796] "IEML's Comparative Advantages," INTLEKT Metadata, https://intlekt.io/iemls-comparative-advantages/ (accessed Jun 22, 2021).

universal semantics. Inspired then by Chomskyan regular languages,[797] Lévy proposes that semantics should be reformulated to be calculable. He proposes a representation of semantic relationships through sets of composable constants and variables.[798] Constants, or semantic primitives, represent the "semantic features shared by all concepts in this semantic domain."[799] Variables, or the IEML Alphabet, are the "range of semantic differences between concepts."[800] Together, these constants and variables can be combined and recombined to formulate meaning.

To then apply the IEML, its building blocks must be further explained. Semantic primitives are the six semantic elements, represented by capital letters, that provide the foundation for the 'metalanguage.' These are: S (sign), B (being), T (thing), U (virtual), A (actual), and E (emptiness). These six elements represent concepts that "empower collective intelligence"[801] and the capacity to make meaning.

The S/B/T operate as a triad. *Sign* is an entity or event that is relevant to knowledge. *Being* is a subject or interpreter and is relevant to the ability to conceive relationships and values. *Thing* is an object or referent capable of categorizing the content. Next, U/A is dialectic. *Virtual* represents the potential or abstract, while *actual* is a "spatiotemporal reality"[802] and represents the tangible or concrete. Finally, E or *emptiness* operates independently and denotes absence, silence, or nothing.

In addition to these semantic primitives, Lévy had created the IEML Alphabet. This Alphabet consists of 25 lower-case letters that when 'multiplied' build various "metaphysical, epistemological, anthropological and existential points."[803] These points, in turn, are understood as "paradigms," or shared semantic relations.

---

[797] The lowest level of the Chomsky Hierarchy, regular languages describe a formal set of grammars that is deterministic. *See* for example "The Chomsky Hierarchy," https://condor.depaul.edu/ichu/csc415/notes/notes10/grammar.html (accessed Jun 22, 2021).

[798] *Id.*

[799] *Id.*

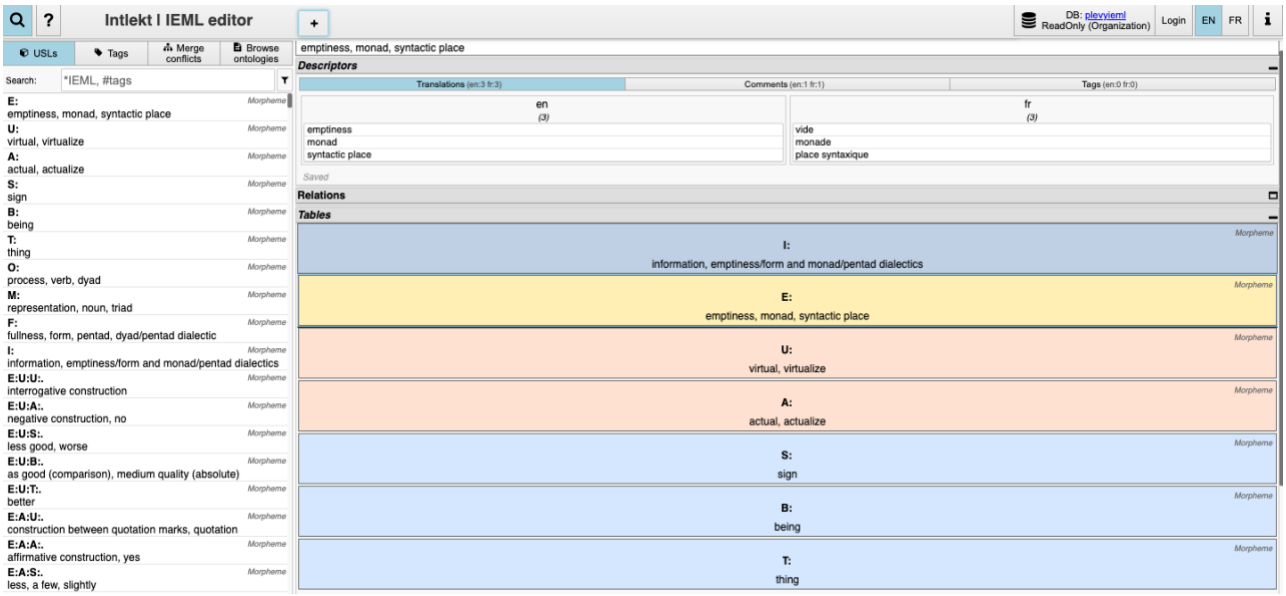
[800] *Id.*

[801] "Semantic Primitives," INTLEKT Metadata, https://intlekt.io/semantic-primitives/.

[802] *Id.*

[803] "IEML Alphabet," INTLEKT Metadata, https://intlekt.io/25-basic-categories/.

Below is a sample of the IEML:[804]



It may be understood, without venturing further, that representation at a new level of abstraction requires defining concepts to a state that is near untenable. Beyond issues of basing its semantic primitives on a constrained set of philosophical traditions,[805] IEML is unintuitive and difficult to grasp. Rather, its competence as a 'universal' semantics[806] can barely capture the nuances of human expression. Consequently, the IEML does little to bridge code with natural language. To unpack semantics to this particular level of abstraction is analogous to using Lego blocks to form a tree. Instead, the exercise should be to reflect on the organic components that allow a tree to grow. In the same way, reconciling code with legal text requires mapping the relations in natural language that have enabled the legal system to persist. This is explored, as we turn to the second half of the chapter.

Computational Legal Inferences and Towards a Pragmatics of Code

In the aforementioned section, it is notable that the problem with using syntax and semantics is akin to the notion of *faux amis*. That is, they do not mean the same things from a linguistic, natural

[804] Captured from the INTLEKT IEML Editor, which allows users to freely experiment with the various computable elements of semantics. See "Intlekt IEML Editor," https://dev.intlekt.io/usl/E:/table/I:.

[805] Lévy relies in a rather piecemeal fashion on loosely Greek philosophy, John Locke, and Cartesianism. Oddly, he draws in some ancient Chinese philosophy, but is not specific about it. *See* section on "Historical and philosophical context for the semantic primitives" within "Semantic Primitives," *supra* 801.

[806] Lévy, furthermore, falls victim to perceptions of a common metalanguage, intercultural language, or an "in-between" language, as capable of working around issues of universal and linguistic grammar. *See* Lin Ma and Jaap van Brakel, *Fundamentals of Comparative and Intercultural Philosophy* 133-139 (2016).

language perspective as opposed to the computable, programming perspective. Their continued treatment as functional equivalents has indeed led to translations that evidently fail to properly capture meaning. However, the insistence of integrating computational technologies in legal drafting suggests that there is, to a certain extent, an inevitability of using programming languages for legal code-ification. So, how could bad translations be avoided, and meaning be reconciled in light of a new medium? To echo Frank Pasquale, "another story is possible."[807] This section aims to uncover the other story by considering first the problem with inference.

In the *Myth of Artificial Intelligence*, Erik J. Larson distinguishes between analysis and formulaic calculation. The former he defines as "making sense of the dots, making a leap or guess that explains them;" the latter he defines as "connecting known dots; applying the rules of algebra."[808] He suggests that "rule-following isn't enough, but it is unclear what exactly else is involved."[809] Larson draws the analogy with murder mysteries and infamous fictional detectives revered for their brilliance in solving seemingly impossible puzzles. He notes that, perceivably, inference from facts is a practice of guessing. Larson references the American logician and philosopher Charles Sanders Peirce, who attempted to map the "mental gymnastics" of Edgar Allan Poe's protagonist, August Dupin, in logical symbols.[810] On method and logic alone, there remains a gap in human reasoning. What may be concluded is that human thought also requires guesswork. The question becomes: how can guesswork be represented?

Larson points to the near forgotten work of Peirce's framework of abductive inference. He suggests that Peirce's thoughts on abductive reasoning remain the missing component to mathematics and logic.[811] More importantly, it persists as one of the reasons that confronts the limits of AI. Peirce distinguishes inference from other forms of thought. Inference is a "leap of sorts, deemed reasonable."[812] Inference depends on some form of prior knowledge and exists in a provisional state. This suggests that the act of inferring encompasses two qualities: (1) context; and (2) incompleteness.

---

[807] Frank Pasquale, *New Laws of Robotics: Defending Human Expertise in the Age of AI* 2 (2020).

[808] Erik J. Larson, *The Myth of Artificial Intelligence: Why Computers Can't Think the Way We Do* 93-94 (2021).

[809] *Id.* at 94.

[810] *Id.*

[811] *Id.* at 99.

[812] *Id.* at 100.

As is the issue with notions of syntax and semantics, inference has frequently been conceived in a broadly singular manner. In conversations about computation and AI, Larson suggests that applications of inference draw largely from a statistical perspective. In effect, he alludes to data-centric approaches and machine learning as analogical representations of inference. There is, however, a distinction between probabilistic inference and inference at an epistemological level. That is, the use of knowledge in context is difficult to capture.[813] This is owed to the exercise of defining relevance. Larson argues that "the ability to determine which bits of knowledge are relevant is not a computational skill."[814]

To then refine the puzzle: if the capacity to infer is uniquely human, what may be the limits of signifying inference computationally? Interestingly, Larson's arguments draw from a systemic perspective of AI.[815] His reflections address how AI systems fail to replicate human thinking. Moreover, he reinforces the point that leaps of faith, paradoxically seminal to scientific advances, were "outside the formalities"[816] and mechanical accounts of practice. Perhaps the most important kernel Larson reveals is that understanding natural language necessitates "commonsense inferences, which are neither logically certain nor (often) highly probable. It requires, in other words, lots of abductions."[817]

Returning to Peirce and guesswork, abduction then involves reasoning that falls outside of logic and leans towards "instinct." While induction draws from facts to build generalizations, abduction is predicated on the observation and speculation of sets of facts.[818] This suggests that explanations and working hypotheses are taken not from facts themselves, but from how they are regarded. Again, information is necessarily partial, contextualized, and incomplete. Aligning inference with conjecture, abduction then regards "an observed fact as a sign that points to a feature of the world."[819] Induction perceives observations as facts, but abduction perceives them as norms. Abductions are

[813] *Id.* at 102.

[814] *Id.*

[815] To a certain extent, Larson refers to artificial general intelligence, which is outside the scope of the thesis.

[816] *Id.* at 103.

[817] *Id.* at 105.

[818] *Id.* at 160.

[819] *Id.* at 163. We consider for example a mirror to Boyd White and the "legal imagination" – how the law sees the world. *See* Boyd White, *supra* 269.

defeasible. Observed facts are clues that operate within a realm of logical possibilities, intentionally including and excluding those on the premise of a specific query.

On the other hand, deduction is "monotonic inference;"[820] conclusions are finite. That is, deductions require that conclusions must be true and that all their premises are true. If even just one of the premises is false, then all the premises are false. Deductive-based approaches are, therefore, dependent on "its truth-preserving constraint –everything must be certain."[821] This means that for deduction to work, the premises must be certain. Consider propositional logic. Truths are derived from propositions.

But what if it is *not* certain whether the premises are true? Could the conclusion still be true? The below set of sentences is an informative example:[822]

> When it rains, the grass gets wet.
> It rained.
> Therefore, the grass got wet.

Though the reasoning here is valid, the premises, and subsequently the conclusion, are not necessarily true. For example, there may have been the *illusion* of rain. A cleaning agency may have been washing the windows of a skyscraper and there was the assumption that the water droplets are indeed precipitation. Or, it rained, but what if the grass is conveniently covered by an awning? This means that even if the conclusion is true, it is not entirely logic; some "luck" is at play. We shall see that, in natural language, it is "impossible to give all necessary and sufficient conditions for the knowledge or application of a concept."[823] The intentional context of natural language, the premises on which inferences are made, can never be completely certain. As a result, though the "basis of correct reasoning is logical deduction,"[824] a theory of meaning is more fundamental and extends beyond logic alone.[825] Monotonic inferences cannot account fully for premises built on *presumed*

[820] *Id.* at 167.

[821] *Id.* at 168.

[822] Reconfiguring Larson's example to a logically valid one. See *id.* at 170.

[823] Ma and Brakel use the informative example of defining "bachelor." While we may be able to specify the necessary characteristics, conditions, such as "unmarried' and "man," there lacks to ability to provide a precise meaning to each of these conditions. That is, we would not consider the Pope to be a bachelor. However, by the conditions alone, he does fit the definition. *See* Lin Ma and Jaap van Brakel, *supra* 806 at 125.

[824] Larson, *supra* 808 at 171.

[825] *Id.* at 170.

certainty. Abductive reasoning, by contrast, introduces possibility, whereby conclusions are not definite. They offer probable conclusions, ones that are the best explanation given a set of premises. Frequently, modal verbs (i.e., may, should, could) act as linguistic clues. Using the above example: when it rains, the grass *may* get wet.

What perhaps is most striking is that models of legal reasoning employ methods of deductive and inductive logic. In a similar manner, computational technologies equally draw from traditions of inductive and deductive modelling.[826] In both scenarios, there has been little reference to the significance of abduction. Yet, conjectural inference is a feature, not a bug, of legal reasoning. Inductive and deductive models, without abduction, is akin to claiming that all law is fact.[827] In contrast, abduction enables the building of analogies; it provides grounds to claim that a horse, or bike, is indeed a vehicle.[828] "Induction requires abduction as a first step"[829] in order to make sense and develop a conceptual framework. Equally, abduction is not an extended form of deduction. As a result, AI systems that reflect either inductive or deductive logic are incapable of wholly reflecting legal practice. Technological advances would only be able to approach, but never replicate legal reasoning.

Peirce's theory of abduction may be extended to the work of John W. Tukey and his argument against the mechanization of inferential knowledge. Tukey was regarded as an atypical member of his scientific cohort. He opposed "rule-bound rationality" and "rigorous objectivity."[830] Instead, he regarded statistical methods as providing clues to "'get a feel' for the data."[831] Often, Tukey described

---

[826] Consider again, for example, data-driven versus expert systems.

[827] The idea put forward by Klaus Guenter that for law to exist, there requires the opportunity for civil disobedience. Normativity is integral to legal systems and the "anarchist feature" is a necessary component. Legal norms depend on social facts, whereas technical rules are mathematical facts. These are two different types of facts with the former akin to custom. Guenter's argument is that 'smart orders' conflate social with mathematical fact, creating systems that are crystallized and incapable of disobedience. *See* Klaus Guenter*, Normative to Smart Orders*, Globinar draft paper (2021). In many ways, this is also a parallel to Latour's discussion on the availability of choice and rules "built-in" to technological systems, effectively forcing compliance. *See* Bruno Latour, "Where are the Missing Masses? The Sociology of a Few Mundane Artifacts," in Bijker and Law (eds.), *Shaping Technology/Building Society: Studies in Sociotechnical Change* 225-258 (1992).

[828] There again, I am alluding to the 1958 H.L.A. Hart "No Vehicles in the Park" hypothetical. There is, in fact, the 1986 case from the Supreme Court of Utah that asked whether a horse was a vehicle under the premise of drunk driving laws at the time. For further detail on the case, see *State v. Blowers*, 717 P.2d 1321 (1986).

[829] Larson *supra* 808 at 161.

[830] Alexander Campolo, "*Thinking, Judging, Noticing, Feeling": John W. Tukey against the Mechanization of Inferential Knowledg*e, 5 KNOW: A JOURNAL ON THE FORMATION OF KNOWLEDGE 83, 85 (2021).

[831] *Id.* at 87.

his work with emotive language to refrain from the scientific hardlines and 'complete truths' that surrounded him. Tukey prioritized observation and was guided by "judgment, experience, and even pluralism."[832] His use of quantitative and computational techniques may be considered as methods of abductive reasoning. Complementary in their arguments, it appears then that Peirce and Tukey illuminate varying strengths in computable analysis, a potentially "weak" form of objectivity.[833]

Interestingly, a parallel may be found in legal theory on the two conceptions of objectivity. George Pavlakos considered the contrast between interpretivism and a discourse theory of law relative to objectivity.[834] He notes that a strong form of objectivity relies on "rigid determinants of truth and correctness."[835] Alternatively, regarding objectivity as a "modest variant" enables an internal reflection on the structures that drive legal propositions. In short, Pavlakos alludes to the type of objectivity found in the discursive legal grammar. That is, discursive grammar embodies "rules that extend over multiple levels of abstraction, as a result of which it can account graphically for the depth of legal practice."[836]

An initial analysis of Pavlakos' arguments reinforces my hypothesis that discussions around computational law must extend beyond the systemic to the micro-level, and specifically to the linguistic space. Therefore, the next step is to reconcile abductive reasoning with the notion of discursive grammar. Abduction is central to understanding the granularity found in natural language, as interpretation necessarily requires both conjecture and defeasibility. This is because natural language is indicative, as opposed to definitive. How natural language signposts meaning is through its grammar. Pavlakos notes that a grammar identifies "'objective' logico-syntactic structure of sentences on the basis of which it is possible to reconstruct the world."[837] The problem, he argues, is one that has been discussed on various occasions in this thesis: the danger of mechanically reducing law to rules. In this case, the rules of grammar replace 'legal rules' that define how law is accurately

[832] *Id.*

[833] Larson notes that Peirce regarded "abduction as a weak form of inference," while Tukey strayed away from an "intense form of objectivity" or mechanical variant. *See* Larson, *supra* 808 at 163 and Campolos, *id.* at 85-86.

[834] George Pavlakos, "Two Concepts of Objectivity," in George Pavlakos (ed.), *Law, Rights, and Discourse: The Legal Philosophy of Robert Alexy* 84 (2007).

[835] *Id.*

[836] *Id.* at 85.

[837] *Id.* at 102.

applied.[838] This has been seen time and time again in computational models, as "grammar" is equated with the likes of syntax and semantics. Consistently, there remains a missing piece: pragmatics.

To recall, pragmatics is concerned with language in use and the contexts of its use. Pragmatics is then primarily focused on implicature and inference: to read between the lines. Interestingly, pragmatics is a subfield of both linguistics and semiotics. Its relevance to the latter will be discussed further in this chapter. For now, it may be notable that Pavlakos' discursive grammar is an excellent starting point. His work actively acknowledges the seminal role of pragmatics. He describes this as the third category of rules that runs alongside the rules of logic and rules of rationality.[839] While semantics reveal sentential logic, pragmatics exposes the normative relations between subjects. In effect, it "opens a gap between the rules of grammar and the criteria for their application, a gap that invites skepticism and indeterminacy."[840]

Consider for example the discussion on implicature and reference in *Language Lego*. Frequently, the use of the pronoun *it* generates confusion around the object of reference. In linguistics, these are pragmatic issues associated either with pronominal anaphora (i.e., pronouns that 'reach back'[841]) or with dummy subjects. Only through context can *it* be identified. Yet, these pragmatic issues remain unsolved computationally. In 2012, Hector Levesque devised Winograd schemas; sets of multiple-choice questions about the meaning of sentences to test for natural language understanding.[842] Winograd schemas demonstrated that machines were incapable of gathering context that extended beyond parameters of syntactic and sentential logic. Below is an infamous example:[843]

> *The town councilors refused to give the angry demonstrators a permit because they feared violence. Who feared violence?*
> a) The town councilors
> b) The angry demonstrators

[838] N.B. *is* as opposed to *should*.

[839] *Id.* at 101.

[840] *Id.* at 102.

[841] Larson, *supra* 808 at 166.

[842] *Id.* at 195.

[843] Taken from *id.* at 196.

Intuitively, it may be gathered from context clues that the pronoun *they* refers to the councilors. For machines, however, the plural pronoun is ambiguous. *They* could refer to either councilors or demonstrators. In this case, the rules of grammar alone cannot resolve pronoun reference.[844] Neither semantic nor syntactic rules can assist with the interpretation of this sentence. On the other hand, pragmatics rules help to signpost meaning. Consequently, the 'grammar gap' Pavlakos describes is akin to Larson's abductive signage. Pragmatics, the linguistic key to abductive reasoning, is integral to knowledge representation, and especially, *legal* knowledge representation.

So, how can pragmatics be represented computationally? I put forth two potential trajectories: (1) using linguistic modelling to blueprint computational models; and (2) programmatically (i.e., the semiotic conveyance of meaning). The first method considers applying core linguistics, specifically pragmatics, as a framework to guide computational strategy. The other draws from critical code studies, an emerging interdisciplinary field concerned with the "extrafunctional significance of code."[845] This is a shift away from interpretation in natural language and towards interpretation in computer code. I make the disclaimer that the methods discussed are not necessarily novel nor can claim to be a comprehensive account. As well, they have been explored in other disciplines (e.g., cultural, media and communication studies). Nevertheless, the significance of the inquiry centers around whether legal text can exist in a form outside of natural language. That is, can computation and code account for the particularities of legal language?

A) Computational Legal Understanding

While advances in machine learning have provided illusions of natural language understanding, there remains an inability to process words with embedded context.[846] Knowledge representation, on the other hand, has largely predicated on logic. As a result, sentence ambiguity (e.g., pronoun reference, polysemy) cannot be completely captured. Attempts at disambiguation have led, instead, to reductive definitions and/or a reframing of concepts.[847] The first and third case studies alluded to the drawbacks

[844] Though Binding Principles in syntax may be informative, they do not provide an explanation when the sentence is already grammatical; only how to generate a sentence that is grammatical.

[845] Mark C. Marino, *Critical Code Studies* 40 (2021).

[846] Recall the discussion in the second case study on translation. *See* Douglas Hofstadter, "The Shallowness of Google Translate," *The Atlantic* (Jan. 30, 2018), https://www.theatlantic.com/technology/archive/2018/01/the-shallowness-of-google-translate/551570/. *See* also Bender and Koller, *supra* 780.

[847] Consider the lessons gathered from the first case study on pigeonholing contractual concepts.

of these computational methods in their current state. Moreover, the aforementioned arguments for abduction further reaffirmed the case study observations. What may be inferred is that, without models of abductive reasoning, there remains limitations in computational representations of law.

Alternatively, the second case study has introduced how legal text may be deconstructed using a combined approach of core linguistic and statistical modelling. Not only has this approach confirmed the significance of interdisciplinary research but has also revealed the need for a multidimensional[848] strategy for computational legal understanding. Furthermore, the outcomes of the case study corroborated prior philosophical interventions that natural language drives legal processes. Consequently, a faithful representation of natural language behaviors is essential to assessing the limits of legal computability. Along this line of thought, a deeper exploration of abductive inference will be conducted.

Though abduction has not held its place in current AI research, this was not always the case. Work on abduction in AI began in the 1970s under the limited context of medical diagnosis.[849] It remained in the realm of medicine until linguists began to conduct research on abduction within informational systems. Their research revealed that, unlike medical knowledge, abductive reasoning for informational systems (i.e., natural language) are, in fact, implicational.[850] In the 1993 seminal paper, "Interpretation as abduction," Jerry R. Hobbs et al. advance a model of abductive reasoning to resolve issues of pragmatics, such as reference resolution. They develop a framework on interpretation that broadly requires two key steps: (1) prove the logical form of the sentence; and (2) make assumptions where necessary.[851] The first step is consistent with existing methods of syntactic and semantic analysis. The second step represents modelling the implicit relations in the sentence; otherwise, the guesswork involved. Hobbs et al. consider that references must be anchored in mutual belief, and that this may be represented in the form of a knowledge base. Consequently, this forms a "referential anchor"[852] that provides information that is presupposed. This is akin to establishing a semantic world and the conditions that make its propositions truths. On the other hand, the second

[848] I define multidimensional as the use of several computational and noncomputational techniques in tandem.

[849] Jerry Hobbs et al., *Interpretation as Abduction*, 63 ARTIFICIAL INTELLIGENCE 69, 117 (1993).

[850] Cf. abductive reasoning in medical knowledge as largely causal. See *id.*

[851] *Id.* at 70.

[852] *Id.*

step involves deriving references from the knowledge base to provide the best guess. This is understood as the speaker's private beliefs.

Consider the following sentence:[853]

> The Boston office called.

In this example, there are three pragmatic issues: (1) the reference "the Boston office; (2) the metonymy[854]"the Boston office"; and (3) the implicit relation between "Boston" and "the office." These three pragmatic problems indicate information that is not defined but inferred from the truth conditions that (a) there exists an office; and (b) there was a call from that office to the speaker. Using a knowledge base approach consistent with their model, the assumption taken is that there is an office, and it is in Boston. As well, the speaker liaises with someone that works in the Boston office. This suggests that this person is referred to as "the Boston office." Moreover, they presumably work in the same office. This is represented, using the linguistic metalanguage, as follows:[855]

that is, B_1 is the city of Boston.

$$office(O_1) \wedge in(O_1, B_1),$$

that is, O_1 is an office and is in Boston.

$$person(J_1),$$

that is, John J_1 is a person.

$$work\text{-}for(J_1, O_1),$$

that is, John J_1 works for the office O_1.

$$(\forall y, z) in(y, z) \supset nn(z, y),$$

that is, if y is in z, then z and y are in a possible compound nominal relation.

$$(\forall x, y) work\text{-}for(x, y) \supset rel(x, y),$$

that is, if x works for y, then y can be coerced into x.

The metalanguage is then situated in a graph shown below:[856]

---

[853] Hobbs et al. use this as an informative example of their model. See *id.*

[854] To recall, metonymy is defined as the thing that is a substitute for the name of a closely related concept. For example, Crown as interchangeable with sovereign or the Queen of England.

[855] *Id.* at 72.

[856] *Id.* at 73.

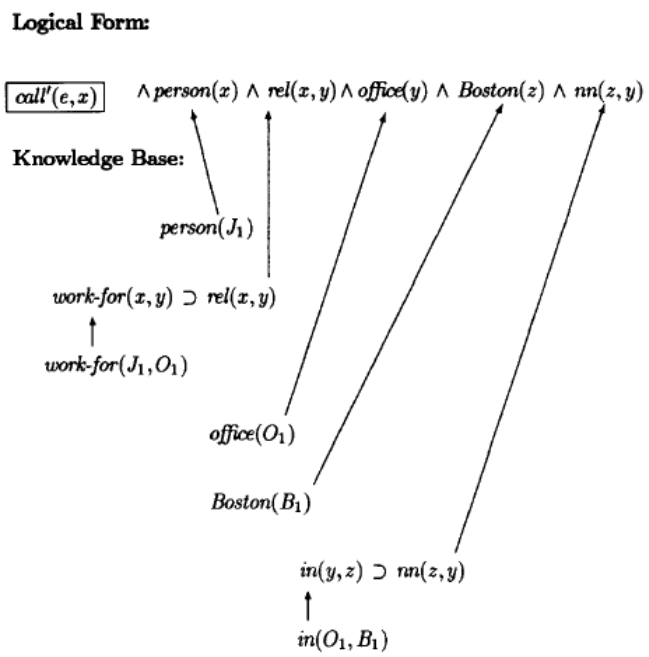


Fig. 2. Interpretation of "The Boston office called."

The combined linguistic and graphical representations, put forward by Hobbs et al., are an early, non-computational model of abductive reasoning. This model then formed the basis of The Abductive Commonsense Inference Text Understanding System (TACITUS), a computational system for interpreting text. TACITUS was constructed on the three pillars of linguistics: syntax, semantics, and, importantly, pragmatics. Accordingly, the system's architecture consists of three components that each correspond with a linguistic pillar. The syntactic and semantic components work through a single system; using a parser to break down the sentence's syntactic structures, then producing a logical form based on "first-order predicate calculus."[857] The logical form then passes through the pragmatics component, "a general abductive reasoning mechanism to uncover implicit assumptions necessary to explain the coherence of the explicit text."[858] In other words, TACITUS reveals the inferences and assumptions required for interpreting text and the coreference relations significant to their interpretation.

TACITUS interprets text by relating the sentence's logical components with the assumptions that can be made. TACITUS tackles several notable pragmatic issues including (1) determining implicit

[857] *Id.* at 75. TACITUS includes a comprehensive grammar of English, enabling predicate-argument relations to be associated with syntactic structures. See also Jerry R. Hobbs et. al, "The TACITUS System," in *Robust Processing of Real-World Natural-Language Texts*, https://www.isi.edu/~hobbs/robust/node2.html (Feb. 24, 2004).

[858] *Id.*

entities and relationships referred metonymically in text; (2) resolving anaphoric references; and (3) expressing relationships underlying compound nominals (noun-phrases).[859] The pragmatic function of the system regards text as "*an instance* [emphasis added] of a schema that makes its various parts coherent."[860] That is, the interpretations of texts require embracing incomplete knowledge. Rather than *the* interpretation, the system highlights *a best* interpretation, and at the very least, *some* interpretation.

TACITUS applies a process known as the "incremental refinement of minimal information proofs."[861] "Minimal information proofs" are regarded as the baseline, whereby a sentence may be understood without context. As domain knowledge grows (through the expansion of the knowledge base), abstract entities and objects in the text are continually "minimized." This means, for example, that objects that share properties are assumed to be identical. This enables possible coreferences for anaphora resolution.[862] Propositions expressed in the text are then related to the other objects known in the knowledge base; in effect, forming an assumption. The intention is to consider interpretation as instances of a number of possible explanations. Assumptions that fit into particular explanations are "preferred to assumptions that do not."[863] As a result, the process is not understood to be definitive. Instead, it is intentionally implicative.

At face value, this may be considered rather similar to inductive reasoning. The difference, however, is particularly highlighted in the representation of *et cetera* in sentences.

$$(\forall x) p_1(x) \wedge p_2(x) \wedge \boxed{etc_5(x)} \supset q(x),$$

Hobbs et al. deliberately include *et cetera* propositions in their knowledge base. *Et cetera* propositions behave as placeholders that associate concepts in sentences.[864] They signal that, to an extent, an implicative relation exists, but is imprecise. While *et cetera* propositions intend to build associations between concepts, they also enable the opportunity to distinguish between objects within

[859] "Robust Pragmatic Interpretation," https://www.isi.edu/~hobbs/robust/node10.html (Feb. 24, 2004).

[860] *Id.*

[861] *Id.*

[862] *Id.*

[863] *Id.*

[864] Hobbs et al., *supra* 849 at 87.

concepts. That is, they liberate implicative relations, allowing an escape valve from absolute definitions.

In relation to legal texts, consider the implications of *et cetera* in legal language. Sandra Fredman describes the "'et cetera' problem," whereby "categories and kinds of subjects can multiply and reconfigure, and how the law can manage such proliferation."[865] Though her argument is a pointed statement around the misuse of *et cetera* in legal interpretation, she brings to light the malleability and potential for growth enabled by such linguistic imprecision. Interestingly, computational systems like TACITUS, preserve indeterminacy, while also allowing implicit references and relationships between concepts to be made more explicit. Consequently, the model put forth by Hobbs et al. is illustrative of the ways in which abductive reasoning can be included in computational law.

Since TACITUS, there has not been a comparable program that has centered on pragmatic processing and abductive inference. As well, the rise of deep-learning and neural networks began to subsume abductive with statistical inference.[866] Syntactic parsers,[867] on the other hand, have since become increasingly powerful owed to advances in deep-learning. Some are even capable of annotating at an incredible level of sophistication.[868] While syntactic parsers have made immense strides in sentential understanding that far exceed TACITUS' logical forms, resolving reference and implicature remain an obstacle. Interestingly, knowledge graph databases[869] have begun to introduce better mappings between conceptual relations. Therefore, further investigation is required in the combined approach of using syntactic parsers and knowledge graphs for the linguistic deconstruction of texts. In this manner, a strong foundation may be laid for an abductive reasoning mechanism. Lessons from TACITUS, as well as the second case study, demonstrate the benefits of using linguistic frameworks as a guide for building computational models. More importantly, developing a computable model of pragmatics will significantly enable a deeper understanding of legal

[865] Sandra Fredman, *Intersectional Discrimination in EU Gender Equality and Non-Discrimination Law* 31 (2016), available at http://ohrh.law.ox.ac.uk/wordpress/wpcontent/up.

[866] Larson, *supra* 808 at 76.

[867] *See*, for example, Stanford CoreNLP. See "Core NLP," https://stanfordnlp.github.io/CoreNLP/ (accessed Jun 22, 2021). Consider, as well spaCy, "Industrial-Strength Natural Language Processing," https://spacy.io/ (accessed Jun 22, 2021).

[868] CoreNLP and spaCy are both capable of managing coreferences, dependency, and other named entity recognition. See *id.* See also "Trained Models & Pipelines," https://spacy.io/models (accessed Jun 22, 2021).

[869] *See*, for example, "Vaticle," https://vaticle.com/ (accessed Jun 22, 2021), as well as "Neo4j," https://neo4j.com/ (accessed Jun 22, 2021).

mechanics. Accordingly, the furtherance of computational law requires infrastructure capable of unpacking the embedded contexts and inherent richness of legal text. Only then can we begin to approach a computational legal understanding.

B) Critical Legal Coding

Stepping outside the realm of natural language, Mark C. Marino proposed that code be read in a manner that extends beyond functionality and the "aesthetic of efficiency."[870] Recall in the third case study on machine-readable legislation, critical code studies (CCS) was introduced as a significant departure from the current understanding of code. Unlike the aforementioned treatment of computation as tools to translate concepts within a natural language paradigm, CCS consider the ways in which code is a system of discourse with its own rhetoric and grammar. Marino suggests that code should not be regarded simply for its reusability and modularity. Instead, this new approach must interrogate the contexts and connotations of the code. He states, "the meaning of code is ambiguous because it is social, even while it is unambiguous because it is technological."[871] Again, this falls outside the typical practices of programming.

The intention of CCS is to be able to read and express code the way "we might explicate a work of literature."[872] It follows that in the process of developing critical hermeneutics, drafting in computer code would allow for a "thickening" [873] of symbolic expressions. Shifting away from its purely functional regard, a turn to the relationships of the code and the choices in programming paradigms could develop "rich methods of reading code."[874] Marino clarifies that he is not echoing the sentiments of literate programming.[875] Alternatively, he is offering the possibility of seeing code as a form of writing that exists beyond operational demands and accuracy.

The case studies have demonstrated the persistent image of code as an emblem of function and practicality. As a result, programming languages were used in a manner that would operate strictly

---

[870] Marino, *supra* 845 at 39.

[871] *Id.* at 40.

[872] *Id.* at 39.

[873] Recall in the idea of thickening as the inclusion of metaphorical and fictional language. *See* Brenda Danet, *Language in the Legal Process,* 14 L. & SOC. REV. 445 (1980).

[874] *Id.* at 41.

[875] Marino references Donald Knuth and his work on "literate programming" and code as communication. See *id.*

on efficiency. This is perhaps owed to a limited regard of the language as strictly syntactic and/or semantic; a focus on structure and outcomes as opposed to content and means. Analogous with learning a foreign language for the first time, code has only been acknowledged in a functional, mechanical sense. Metaphor, irony, fiction, and other complex uses of language have not been considered because code has yet to be perceived as worthy of interpretation. In defining, then, techniques of critical analysis, the potential of code, as a non-natural[876] but linguistic medium, will be tested against the requirements of legal language. In doing so, I aim to make a preliminary assessment on the prospect of legal codex(t).

Marino raises Douglas Hofstadter's notion of meaningful isomorphisms, the "relationships drawn between one system and another."[877] Marino's discussion of isomorphisms significantly points to the misnomers and *faux amis* between computer science and law. Under Hofstadter's definition, isomorphisms fall closely in line with "transliterating;" otherwise, matching the concepts of one language directly to the other.[878] This notably has been problematic, as according to Hofstadter, meaningful isomorphisms necessitate that the systems in question be completely interchangeable. Evidently, this is not the case between legal and computational systems, nor between natural and programming languages. The truths of one system are not necessarily the truths of the other.[879] I consider that the 'isomorphic technique' and practice of matching has been the predominant approach used in Legal Tech. More importantly, this matching presupposes that natural and programming languages operate on the same semiotic paradigm. Marino, therefore, recommends a relational method: to identify connections between the sign and their referents, and the forces that shape their meaning.[880] In this manner, Marino suggests that code must be interpreted for its gestures and performance. In other words, a pragmatics of code must be considered.

Marino sets out several practices for CCS and interpretation using this relational method. First, the use of code must be perceived only as an "entry point to an investigation."[881] He argues that every

[876] To recall, this is to note that between signified and signifier, it is not an obvious connection. *See* Betty J. Birner, *Language and Meaning* (2018).

[877] See discussion *id.* at 42. For full detail on isomorphisms, *see* Douglas Hofstadter, *Gödel, Escher, Bach: An Eternal Braid* (1979).

[878] *Id.*

[879] *Id.*

[880] *Id.*

[881] *Id.* at 48.

piece of code is incomplete. The existing task-based understanding of code has led to a misguided assumption around the context-independence and determinacy of lines of code. Though code is frequently removed from its development environment and transposed across systems, platforms have emerged to enable users to import code that identifies their source code repositories.[882] This allows the code to remain "connected to their context" with comments on the code possibly made "in situ."[883] An analogy may be drawn to quotations or citing in natural language, enabling a form of textual grafting. Though the sentence may be displaced from its original text,[884] and in effect, foster a new meaning, there remains the option to trace back its history and social origins. This suggests that code is not context-independent nor determinate, but, rather, capable of effecting meaning in illimitable contexts.

Second, the choices around the specific combinations of code must be analyzed. As opposed to assessing whether they are valid lines of code, its purposeful arrangement must be accounted. Indeed, code can present "signs of 'humor, innovation, irony, double meanings, and a concentration on the play of language.'"[885] The arrangements of code can be aesthetic. Consider the following excerpt of code:[886]

For instance:

```
$walk1_beat = ++$walk1_beat % 16;
```

One might add parenthesis to make this clearer, or not.

```
$walk1_beat++;
if ($walk1_beat eq 16) { $walk1_beat=0 }
```

Both are capable of executing the same output. However, in the latter, the use of '`eq`,' rather than '=,' is a subtle play on meaning. Though functional equivalents, the former is used to compare

[882] Marino uses the example of the ANVC Scalar software platform that allows the importing code as text from source code repositories. *Id.* at 49. *See* also in discussion on Scalar features: "not only can any piece of Scalar content become a path or tag (or both), but it can also reference any other piece of content." *See* "Flexible Structure," About Scalar, https://scalar.me/anvc/features/flexible-structure/ (accessed Jun. 20, 2021).

[883] *Id.*

[884] Consider the reflection from Derrida and deconstruction.

[885] *Id.* at 49. Marino cites Loss Pequeño Glazier, "Code as Language," *Leonardo Electronic Almanac* (2006).

[886] Example from Marino. See *id.* at 50.

strings, while the latter numbers. It follows that the valid/invalid binary parallels only grammaticality judgments in natural language. It does not factor stylistic intention. Moreover, how the code attempts to perform has imprints of its epistemologies, cultural, and political paradigms.[887] Code communicates through its symbols and whitespace.

In the "Aesthetics of Generative Code," Geoffrey Cox et al. advance the notion of a "poetics of generative code."[888] That is, the value of code is only revealed at the time of execution. They note that the code, frequently 'read' and referenced, is only its written form. This mistakenly reduces code to mere machine-readable notation and implies that code is limited to expressions of logic. In effect, this falsely conflates form with function. Alternatively, they argue that to build proper criticisms of code, one must also understand the code's actions. Code does not operate in a single moment in time and space, but as a series of consecutive actions that are repeatable.[889] Outcomes then are capable of imagination in different contexts.

Importantly, the effects of the written code are not known until its execution. A comprehensive literacy of code enables plays on its structure; to use distinctive syntactic operators to produce a specific arrangement.[890] Yet, the code's execution is its chronotope.[891] It materializes the abstract elements and particular design choices in the arrangements. It is where meaning and narrative of the code is bridged with its makeup. Its reality then is remade and redescribed, a suspension of the direct description to the metaphorical one.[892] Code is shaped by its performance. Subsequently, the analysis of code should consider its constant shifts in state. As discussed by Cox et al., code has an interesting temporal relationship. The written expression of code – or it's static form – "represents a form of its

[887] *Id.* at 50.

[888] Geoffrey Cox, Alex McLean, and Adrian Ward, "The Aesthetics of Generative Code," *International Conference on Generative Art* (2000).

[889] *Id.* at 8.

[890] *Id.* at 6-7.

[891] To use Bakhtin's term, chronotope, defined as "the points at which the knots of the narrative are tied and untied [...] and emerges as a center for concretizing representation. *See* Mikhail Bakhtin, *Dialogic Imagination: Four Essays* (1981).

[892] In reference to Paul Ricoeur as he describes narrative as the "world of the text that intervenes in the world of action in order to give it a new configuration or, as we might say, in order to transfigure it." *See* Paul Ricoeur, *From Text to Action* 10-11 (1991).

existence before it is processed by the machine."[893] The reading of code, then, requires moving past its static form to understand the effects caused by symbols during its dynamic engagement.[894]

Code must be understood in action; only then are design choices situated and contextual references revealed. To interpret and develop critical hermeneutics, code must be understood holistically: beyond programmatic syntax and semantics to pragmatics. Marino argues, code "yield[s] meaning to the extent to which we interrogate their material and sociohistorical context, [...] and read their signs and systems against this backdrop."[895] Consequently, code must be read against the backdrop of its own context vis-à-vis its transposed one.

In applying the practices of CCS, code is undeniably a form of writing.[896] More importantly, its interpretative practices illustrate that while code is not isomorphic to natural language, code as text is not inconceivably different from natural language text. Some overlap exists. The test, however, is not whether text *generally* is inclusive of code. Rather, the test is whether legal text could be code; in effect, a legal codex(t). In *The Linguistic Affair*, the literature has revealed that the legal language is rather distinct. Moreover, legal concepts have relied on natural language for their expression. Yet, it remains unclear whether natural language may be the only form of legal writing. That is, can legal writing exist outside of natural language construction?

Reflecting on the distinctiveness of legal language, the initial task is to determine whether code could fulfil the demands of the language. Recall the unique behaviors that distinguish legal language from others. Peter Tiersma acknowledged the oft-arcane qualities of the technical language, but, nevertheless, asserts that both the lexical and structural complexities are intentional. Rather, the language is not merely communicative. Its stylistic form is not embellishment, but in fact, integral to its function. That said, what Tiersma alludes to is the law's conceptual complexity traceable through its linguistic patterns. Other scholars, such as Brenda Danet and James Boyd White, have noted that these stylistic choices represent the symbolic significance and ritualistic behavior of the language.

---

[893] Marino, *supra* 845 at 51,

[894] *Id.* This is ever the more apparent in Ricoeur that between understanding and explanation is observed in the domain of poetics. He describes how the act of understanding requires "grasping the semantic dynamism by virtue of which, in a metaphorical statement, a new semantic relevance emerges from the ruins of the semantic nonrelevance as this appears in a literal reading of the sentence." *See* Ricoeur, *supra* 892 at 9.

[895] *Id.* at 53.

[896] Conceivably, we may be able to draw parallels with Latour on text and artifacts as organizing the "relation between what is inscribed in them and what can/could/should be pre-inscribed in the users." *See* Latour, *supra* 827 at 237.

The poeticism of legal language, reinforced by literary devices of metaphor and fiction, is instrumental to its existence. The legal language is perceivably figurative and requires it to be experienced. It is a specific imagination of fact and configures narratives as truths. As well, the legal grammar reveals the law's "strange retrospective temporality."[897] Neither causal nor chronological, legal language establishes commitments made in the present, for the future, by referring to the past. This non-linear interpretation of time is an implicit representation of the incompleteness of law, its knowledge is interruptible and incapable of total attainment.

Broadly, the legal language may be categorized by three distinct markers: (1) conceptual complexity; (2) poeticism; and (3) temporal specificity. Conceptual complexity describes the innate use of specific vocabulary and peculiar sentence constructions for the communication of legal concepts. Poeticism reflects the use of literary device and the heavily figurative quality of the language; and, finally, temporal specificity articulates the law's particular relationship with time. Again, applying the aforementioned CCS practices as a framework for code's 'textual' competence, preliminary observations suggest that code appears to conform with the demands of the legal language.

The CCS practices reveal that code is conceivably (1) incomplete; (2) poetic; and (3) temporally driven. The second and third traits seem rather self-evident. That is, there are demonstrably artful manipulations of syntactic operators that enable duality of meaning and metaphorical representation. As well, the portability of code to different platforms can equally be situated with their original contexts. This fosters a better understanding of their "sources." Code is also sensitive to its dynamic engagement, highly mutable and susceptible to change. Together, these two traits pair well with the second and third characteristics of legal language.

The first trait, however, is more complicated and perhaps the crux of this investigation. It places at the forefront whether the lexical and syntactic complexity is inherent to the law's performative character. Recall the lessons drawn from Danet's study[898] on conceptual and linguistic complexity in legal language. Her observations reveal that increased conceptual difficulty does not necessarily lead to reduced comprehension. But, neither does lexical nor syntactic simplification. Again, this means that clarity and simplicity are not synonymous. Furthermore, this runs into the problem deconstructivism presents and, more broadly, the Sapir-Whorf Hypothesis. That is, language affects

[897] Referencing Marianne Constable, *Law as Language*, 1 CRITICAL ANALYSIS OF LAW 68 (2014).

[898] *See* in *The Linguistic Affair;* Danet, *supra* 873 at 488.

conceptions of reality. In this case, natural language affects —and has affected—conceptions of law. Legal complexity is intrinsic and cannot simply be resolved. So, is this the end of the path? How then could code be reconciled as legal writing?

The current difficulty with 'code-ification' may be described as forcing square pegs in round holes. It is an attempt to draft computational legal expressions by extracting the underlying logic of legal processes. This, in turn, flattens and compresses the richness of law. Moreover, it assumes that legal norms may be 'transferred' from one container to another. In contrast, accepting that natural language has already impacted the construction of legal concepts, only one criteria of evaluation is relevant. That is, code should only be assessed for its ability to inherit natural language's traits. The most fundamental being indeterminacy. Should the indeterminacy of the law reflect the indeterminacy of the language, then code should simply be tested for its inherent incompleteness. In that regard, code is indeed indeterminate. Code is ambiguous. Code is partial.

Nevertheless, the inquiry becomes: what is the benefit of drafting in code as opposed to natural language? Why should code even be considered legal text? The literature review and case studies have shown that arguments for legal code-ification typically fall in line with simplification and efficiency. In fact, the argument should be one of clarity and accessibility. David Mellinkoff was perhaps first to conflate clarification with simplification. This has dangerously implied that legal complexity should be reduced. Evidently, attempts at simplification have accomplished what has been akin to reckless extraction and bad translations (i.e., transliterating or decoding). A hurdle experienced most presently in discussions around a domain-specific language for law. On the other hand, it has been demonstrated that, overriding paradigmatic shifts, or reconceptualizing entirely away from natural language, runs into problems of overcomplexity.[899] How then could natural language maintain its signature[900] in code?

Interestingly, CCS has provided a fascinating illustration of how code can inherit and retain its natural language ancestry. Consider the command `PRINT`. Marino describes the various evolutions of the term. Historically, printing began as the notion of putting words on paper (or, parchment).

[899] Recall IEML and the complexity with "semantic interoperability." *See* Lévy, *supra* 796.

[900] As a reference to Giorgio Agamben of signatures as "archaeological traces." Recall Giorgio Agamben, *The Signature of All Things* 36 (2009).

Importantly, *print* has come to signify a "system of inscription."[901] The word *print* itself "bears no automatic relationship to what [it] stands for."[902] It is arbitrary. In programming languages, `PRINT` is understood as the display of data on the screen. Just as most linguistic meaning, programming commands and variables may be represented using any select combination of characters. `PRINT` could just as easily be `TNIRP`. The intentional choice of `PRINT` represents a continuity in humanistic tradition, history, and sociopolitical origins.

Likewise, inherent to the legal language is a preservation of tradition. Though Mellinkoff may regard it as "weasel words,"[903] the persistent use of archaisms (i.e., Middle and Old English, Latin and French) reflects the same form of continuity. Therefore, legal codex(t) is conceivable to the extent that it inherits its natural language roots and embodies existing complexity. Moreover, there must be mechanisms in place for the legal language to refer between the analog (natural language) and the digital (code). The legal language must continue to be seated within a network of its history, relationships, and evolving contexts. In this way, the integrity of legal norms is maintained, and human-centricity is upheld. It follows that an associative code for legal writing is premised on establishing first computational legal understanding – in effect, an infrastructure for clarifying legal knowledge.

Importantly, there is a significant difference between translation and drafting. To imagine a legal codex(t) is not to frame it as a question of translation. Instead, it is a reflection of whether code has the capacity to draft going forward. Interestingly, Lexon had provided a pioneering effort on the use of natural language constructions as executable code. However, this ran into issues of reconceptualization, asserting of their own framework to existing legal interpretations. This suggests that, rather than re-writing existing legal texts in code, the exercise should be one of reference.[904] It requires applying knowledge attained from computational legal understanding to develop this associative code for legal writing. It is the formation of a computational legal network.

---

[901] Marino, *supra* 845 at 42-43.

[902] Birner, *supra* 876 at 4.

[903] David Mellinkoff as he references Stuart Chase, *The Tyranny of Words* 324 (1958) in *The Language of Law* (1963).

[904] I define *reference* here as belonging to relational knowledge, a mirror to the relational conception of law. I refer again to Hildebrandt on the law's existence as dependent on the "performative nature of the social fabric it constitutes and by which it is constituted." This is specified in the relationship between information and communication infrastructures and law. *See* Hildebrandt, *supra* 4 at 172.

Concluding Remarks

Undoubtedly, the ideas put forth require further examination. For now, it may be important simply to acknowledge that pragmatics has been, and continues to be, a missing piece to the Legal Tech puzzle. Current uses of programming languages and computational technology have made strides in 'clarifying' the law through simplification. This method, however, treats complexity as a defect and is revealed in the persistent focus on syntactic and semantic techniques of legal knowledge representation. Again, this is not to suggest that logic and structure is not part of the equation, but that it is not the entire solution. Instead, the richness of the law should be preserved through methods of representing pragmatics computationally. This extends into perceptions of code. That is, code should be critically analyzed for its interpretative potential beyond function. In doing so, can benefits of quantitative method be bridged with normativity; thereby reintroducing the space for argument and indeterminacy. Nonetheless, the limitation persists in how a code's own ancestry and system of norms may be reconciled with legal norms.

EPILOG(UE)

Twenty years since "Aesthetics of Code," Geoffrey Cox et al. had continued on the trajectory of defining a new paradigm of code work. In 2004, Cox et al. had written a response to their original paper, further arguing for a framework to produce code that encapsulates a critical practice.[905] In 2012, Cox, along with Alex McLean, published this framework in their book, *Speaking Code*. Most recently, Cox collaborated with fellow software studies and computational practices scholar, Winnie Soon, on *Aesthetic Programming*. I will briefly summarize the aforementioned texts to offer perspective on the emerging horizons of code as critical and literary scholarship. I consider, as well, how these methods may be relevant for legal codex(t). Furthermore, I hope to illustrate that, beyond aesthetics, code as legal expression is not merely speculative but may, in fact, be on the rise.

Cox et al. state "the formal qualities of code cannot be separated from its broader discursive framework."[906] In the prior chapter, this has been clearly described in the misperceptions of code as merely its notation of logic. Code, however, is only understandable within the context of its overall structure.[907] That is, though the components may be predetermined, "the combinations of interactions combined with the dynamism of unpredictability"[908] result in its incompleteness. Coding requires human intervention; code is speculative. Moreover, code is imperfect, as it is subject to mistakes that could alter the course of its performance. Code is in a continuous state of 'becoming.'[909]

Interestingly, Cox et al. describe how programming follows closely with abductive reasoning. Programmers frequently take "leaps of faith" in their process.[910] This is owed to code being capable of self-modification. This means that there is an extent to which programmers can only anticipate how code can function, as the code itself can modify its own behavior. Self-modifying code then "breaks the determinism of code and makes it explicit."[911] Therefore, to understand code necessarily involves unpacking its embedded theory applied to practice. The theory, of course, reveals the intrinsic nature of code as a linguistic practice.

---

[905] Geoffrey Cox, Alex McLean, and Adrian Ward, "Coding Praxis: Reconsidering the Aesthetics of Code," in Olga Goriunova and Alexei Shulgin (eds.), *read_me, Software Art and Cultures* 172 (2004).

[906] *Id.* at 162.

[907] *Id.* at 164.

[908] *Id.*

[909] *Id.* at 167.

[910] *Id.* at 171.

[911] *Id.*

In the foreword to *Speaking Code*, Franco Berardi articulates that code has the power to "inscribe the future, by formatting linguistic relations and the pragmatic development of algorithmic signs."[912] What he describes is, in effect, advancing towards a pragmatics of code. Having confounded code as "syntactic exactness of linguistic signs," Berardi suggests that through "excess," or poetry, are the limits of the signified reopened.[913] In short, we are encouraged to redefine the limits of code: to interpret code as writing. With great intention, text and code are interplayed throughout the book to underscore that code is indeed text. Moreover, Cox and McLean shift away from the "reductive tendencies" in machine reading to acknowledge that code is an "active agent" in the process of meaning production.[914] They argue that once code is likened to speech, then natural and artificial languages may be combined to develop new meaningful speech acts. Coding is "a mode of action," in which "ideas are stated and then reflected upon and restated."[915] But, code differs from other forms of writing; in the sense that it must follow quite literally its script. As a result, its predeterminations are paradoxically also its "sense of excess."[916] The poetry is inherent to its practice.

Accordingly, coding practices follow a few core principles that are beyond "simply the demonstration of formal logic."[917] The most important is the notion of double coding. They argue that "codework," or what is occasionally referred to as pseudocode, introduces meaning that is seemingly prescriptive but is non-executable.[918] Pseudocode is a design tool for the description of the code and uses the structural conventions of the programming language. It is intended for the human to read, and not the machine. Pseudocode does not have any formal impact on the executable code but is significant in defining how the code may be implemented. Moreover, it is a representation of the code. Consequently, double coding suggests that pseudocode puts forth a "double sense of interpretation."[919] In effect, it acknowledges the ambiguity that may arise owed to the potential divorce between design (intended meaning) and implementation (actual meaning).

---

[912] Geoffrey Cox and Alex McLean, *Speaking Code: Coding as Aesthetic and Political Expression* ix (2012).

[913] *Id.* at xii.

[914] *Id.* at xiii.

[915] *Id.* at 14.

[916] *Id.* at 11.

[917] *Id.* at 8.

[918] *Id.*

[919] *Id.* at 9.

Equally, Cox and McLean consider "secondary notation" as a core principle. Secondary notation is inclusive of coding practices, such as "commenting out" or the choice of variable names and/or identifiers.[920] In the former, placing a "#" denotes that what follows is not part of the source code. These comments are then excluded from the actual execution. As we've seen with the latter, naming variables in code also does not have an impact on execution.[921] To the computer, variable names have no meaning. Interestingly, secondary notation pejoratively suggests that 'reading' done by machines is the code's primary practice. In contrast, Cox and McLean argue that secondary notation maintains the human aspects of the code.[922] In fact, it plays an important role of integrating the author's voice to the code. Secondary notation then fosters the intentionality and purpose behind the code.

Consider the "codework" written in the Perl programming language that interplays secondary notation with executable code:[923]

```
package DONT::CARE;
use strict; use warnings;
sub aspire {
my $class{tab}          = POOR;
my $requested_type      = GET_RICHER;
my $aspiration{tab}     = "$requested_type.pm";
my $class{tab}          = "POOR::$requested_type";
require $aspiration;
return $class->new(@_);
}
1;
```

Notably, the programmer, Graham Harwood, provides a commentary on social and economic stratification. The term "class"[924] is double coded to "stress the material conditions of working with

[920] *Id.* at 23.

[921] *Id.* Recall as well the discussion by Marino on the evolution and use of the `PRINT` command. *See* Mark C. Marino, *Critical Code Studies* 42-43 (2020).

[922] *Id.*

[923] Cox and McLean take the extract from Harwood's codework *Class Library* (2008). See *id.* at 40.

[924] To recall, this is a term used in object-oriented programming to describe one or more objects in the code.

code against labor conditions and class struggle."[925] This interplay between the secondary notation and the executable code, together, reflects how code practice could extend normative perceptions on socioeconomic conditions. More importantly, in recognizing that code can account for the "dynamic character of social processes,"[926] and can embody both linguistic and communicative mannerisms, deterministic conceptions of code are broken down. This has particularly significant implications as secondary notation had been considered in the guise of computational contracts. I will return to this later in the section.

Further ambiguities that arise in coding practices include the use of syntactic operators like "or," "and," "not," as well as infinite loops.[927] Similar to logical connectives in core linguistics,[928] certain syntactic operators extend beyond its 'grammatical' use. In contrast to perceptions on context-free grammars, structural elements in code provide context clues and is discursive.[929] Consider for an example a loop. In programming languages, loops provide instructions and conditions for when certain actions are to be repeated. As well, loops may be nested within loops, signified via parameters "{ }". The placement of loops establishes the points at which sentences should be subclauses. This is analogous with the strategic use of logical connectives. Its meaning is only conveyed when reading the text as a whole. Moreover, loops challenge the "conventional structures of linear time."[930] The inclusion of certain loops "mirrors the complexity of lived time" and represent the experience of it.[931] Again, the arrangement of the code, how it is organized, is deliberate and serves not function, but stylistic intention.

However, meaning in natural language can draw from the subconscious, while systems of meaning in code are primarily conscious. This is not to be confused with the act of making code more explicit. For Cox and McLean, this means that code 'augments' existing relationships by compiling various

[925] *Id.*

[926] *Id.*

[927] Infinite loops are coding instructions to repeat an action indefinitely. They often structure the program, but the use of infinite loops paradoxically comes with the possibility of threatening the logical structure. See *id.* at 10.

[928] Recall discussion in *Language Lego* on logical connectives and pragmatics.

[929] Cox and McLean, *supra* 912 at 20.

[930] Winnie Soon and Geoff Cox, *Aesthetic Programming: A Handbook of Software Studies* 91(2020)

[931] *Id.* at 92.

models of human perception. This is furthered by the composability of code[932] and is comparable to compositional meaning in semantics.[933] That is, how components are woven together and built up have an impact on the overall meaning of the text. As discussed previously, code is capable of behaving like building blocks that can be displaced and reassembled in different environments. Consequently, entire code systems may be embedded with one another, producing meanings that are deeply interwoven. Code, therefore, exists as "part of wider social relations"[934] that already embody systems of societal norms. While this should be distinguished from the grammar associated with foundations of coding,[935] the question remains how code's own system of norms can be reconciled with legal norms.

In *Aesthetic Programming*, Soon and Cox experiment with code literacy by weaving together "the words and actions of human and computer languages."[936] While it is considered a handbook, its intention is to address the "more complex and deeply entangled set of relations between writing, coding and thinking."[937] That is, they consider the practice of building and making worlds by relating fundamental programming concepts with political paradigms and their power relations. Soon and Cox describe this as "expanded literacy," an "enhanced understanding of the relationship between what words mean and do in terms of wider culture."[938] Though they are sensitive that code is not a natural language and is not conceivably equivalent, they stress the significance of code as a linguistic medium, capable of providing expression through its own form of semantic ambiguity.[939] As a result, they expand on the analysis of secondary notation, particularly in the naming of computational objects and functions.

Two sections, in particular, will be highlighted: (1) object abstraction; and (2) vocable code. I have elected to consider these sections, as they best capture the qualities significant to legal language and

---

[932] *See* for example Linda Xie, "Composability is Innovation," *Future* (Jun. 15, 2021) https://future.a16z.com/how-composability-unlocks-crypto-and-everything-else/.

[933] Recall in *Language Lego* in semantics.

[934] Cox and Mclean, *supra* 912 at 27.

[935] To clarify, I am referring to the various practices associated with programming basics as opposed to the embodied social paradigms.

[936] Soon and Cox, *supra* 930 at 45.

[937] *Id.* at 13.

[938] *Id.* at 44.

[939] *Id.* at 45.

legal knowledge representation. Introduced in the second case study, object-oriented programming (OOP) finds striking intersections with legal reasoning. To recall, OOP is the structuring of code as objects, rather than logic.[940] This means that OOP is a form of managing complexity through abstraction, and in effect, concretizing it. Therefore, it speaks heavily about representation. Soon and Cox note that, in the practice of object abstraction, attention must be turned to the subjectivity involved in the movement between abstract and concrete reality. It requires understanding the "hidden layers of operation and meaning."[941] This process of reducing complexity is likened to "desktop metaphors."[942] Though they are capable of increasing accessibility, there must also be an acknowledgment that simplification is not a neutral exercise.

Computational objects are constructed by selecting properties and behaviors that are perceivably important in their representation.[943] Others are ignored, fostering the "suppression of a lot of other aspects of the world."[944] Crutzen and Kotkamp note that abstractions create "illusions of objectivity"[945] when representing the complexity of processes and its relations. This is because the design reflects highly organized imaginations of the world, specifically as independent objects that operate and interact with one another.

OOP can then be understood as a "configurative system of discrete, interlocking units of meaning."[946] Not only is this reminiscent of the aforementioned notion of composability, but also alludes to ancestry, the inheritability of traits, and the network of "interlinking agencies."[947] Put differently, OOP draws attention to relationships between entities and analog understandings of them from abstract grouping and categorization. More importantly, it suggests that there is little difference to the processes of relaying abstract concepts in "analog" practices. Akin then to placing deleted files in 'trash bins,' the significance of OOP stems from its ability to reflect complex processes with

[940] *Id.* at 145.

[941] *Id.* at 146.

[942] Soon and Cox allude to the analogy of deleting a file as throwing it in the trash bin. See *id.* at 145.

[943] *Id.* at 147.

[944] Soon and Cox cite Cecile Crutzen and Erna Kotkamp, "Object Orientation" in Matthew Fuller (ed.) *Software Studies* 202-203 (2008). See *id.* at 147.

[945] *Id.*

[946] *Id.* at 160.

[947] *Id.* at 161. *See* also the notion of "actants" from Bruno Latour, "On actor-network theory. A few clarifications plus more than a few complications," available at: http://www.bruno-latour.fr/sites/default/files/P-67%20ACTOR-NETWORK.pdf.

familiarity. OOP, therefore, reinforces the argument that for a legal codex(t), there must first be a continuity of conceptualization. This is continuity must be informed by legal constructions in natural language. Subsequently, vocable code may be informative of the manner in which text and code are interwoven.

Vocable code is a play on secondary notation and considers the performativity of code. It emphasizes how code "mirrors the instability inherent in human language in terms of how it expresses itself, and is interpreted."[948] In understanding the instability of code, it is then possible to recognize how particular meanings may be "open to misinterpretation and reinvention."[949] Importantly, vocable code does not regard the prospect of misinterpretation as a flaw, but simply as an attribute. Vocable code is an elaboration on the existing framework put forth by Cox and McLean in prior literature (i.e., *Speaking Code*), transformed into a programming method. This framework highlights Derrida's notion of writing as marked by absence.[950] It wrestles with the gap left by the 'voice' of the author for the 'voice' of the prospective reader. The interest of the source code is to "blend form with function."[951] The source code sends instructions to machines, while also communicates with humans. One of the key practices to vocable code is "constraint-based writing."[952] This is understood quite simply as writing program code with certain rules.[953] However, these are stylistic rules, intended to 'undo' the usual way of writing code, "such as not using the single x and y, one and zeros as integers, true and false Boolean, or the single operator of > or <. The source code does not prioritize efficiency [...]"[954] Therefore, this practice represents the duality of combining formal logic with poetic expression; that even syntactic constraints can be intentionally normative and discursive.[955]

A few conclusions may be drawn from aesthetic programming. First, the imagining of code as writing reaffirms a fundamental argument of this dissertation. Namely, that code, like legal language, is a

[948] *Id.* at 167.

[949] *Id.*

[950] Recall again Jacques Derrida and deconstruction in *The Linguistic Affair*.

[951] Soon and Cox, *supra* 930 at 168.

[952] *Id.* at 169.

[953] For further detail on constraint-based writing, *see* Eva Heisler, "Winnie Soon, Time, Code, and Poetry," *Asymptote Journal* (Jan. 2020) https://www.asymptotejournal.com/visual/winnie-soon-time-code-and-poetry/.

[954] *Id.*

[955] The core method for structuring vocable code is to use very specific constraints on its structure. Yet, they may be equally discernible for its meaning. See *id.*

social phenomenon that inherits meaning through historical and institutional legacy.[956] As a result, legal codex(t) necessitates preserving a network understanding of both the internal order and relationship to other discourses. This is demonstrably possible through OOP as well as code's inherent composability. Second, for there to be a successful "grafting," code as legal expression must be able to uphold its conceptual continuity. To do so, there must be a reevaluation of object abstraction and secondary notation.

Interestingly, there are emerging prospects in this regard. AuthoritySpoke, developed by Matt Carey, is both a platform and set of tools that work with three forms of legal data: (1) court opinions; (2) legislative enactments; and (3) legal procedural rules.[957] Using Python classes, AuthoritySpoke employs an OOP to represent various aspects of legal reasoning. Importantly, AuthoritySpoke does not intend to 'translate'[958] legal language. Instead, its goal is to provide computational annotations of existing text.[959] These annotations overlay legal documents and are designed to help clarify legal concepts. Moreover, it preserves both the technical and legal ancestry by using (1) existing Python programming patterns; and (2) the same natural language phrasing to articulate legal concepts. This offers a method of reconciling technical with legal norms.

Consider the excerpt from AuthoritySpoke's technical documentation:[960]

> Predicates can be compared using AuthoritySpoke's `.means()`, `.implies()`, and `.contradicts()` methods. The `means` method checks whether one Predicate has the same meaning as another Predicate. One reason for comparing Predicates using the `means` method instead of Python's `==` operator is that the `means` method can still consider Predicates to have the same meaning even if they use different identifiers for their placeholders.

[956] Refer to Peter Goodrich, *Legal Discourse: Studies in Linguistics, Rhetoric and Legal Analysis* 144-151 (1985).

[957] "An Introduction to AuthoritySpoke," *AuthoritySpoke* https://authorityspoke.readthedocs.io/en/latest/guides/introduction.html (accessed Jun. 22, 2021).

[958] In the meaning consistent with *Weaving the Code*.

[959] AuthoritySpoke explicitly does not intend to turn Python into logic programming nor designed as a deep-learning model. *See* "Using Python Template Strings to Represent Legal Explanations," *Python for Law* (Jan. 22, 2021) https://pythonforlaw.com/2021/01/25/python-template-strings.html#h-higher-order-predicates. *See* also "Using Python Template Strings to Represent Legal Explanations," *AuthoritySpoke* https://authorityspoke.readthedocs.io/en/latest/guides/template_strings.html (accessed Jun. 22, 2021).

[960] *Id.*

Observably, Carey applies a form of "constraint-based writing," as previously described by Soon and Cox. This is revealed in the explanation on the use of "`means`" rather than "==." The added method, as opposed to the syntactic operator, is exemplary of critical coding. The choice to use a "`means`" function highlights that the code is sensitive to the possibility of multiple meanings. Furthermore, the function does not ascertain a particular meaning, but rather highlights a relational connection between one or more entities. Also, it is notable that these functions are based in predicate logic. What may be gathered, again, is the marriage of logic and poetic expression. Though code operates within logical structures, it is, nonetheless, discursive. The AuthoritySpoke documentation offers many other examples (i.e., implicature,[961] temporal reference[962]) worthy of further exploration. It must be disclaimed that they are currently in their infancy and are continuously adding new functions to their platform. Still, a preliminary look into their code work illustrates the rising potential of legal codex(t). An area that remains outstanding is how AuthoritySpoke may be able to capture legal fictions.

Another fascinating prospect may be a re-evaluation of programming languages for contracts. To recall, I reflected on the legal effect of annotations in certain formal languages (e.g., Solidity, Sophia, or Lexon). A 'quick fix' that was proposed was to give legal authority to these annotations. This approach was suggested by Shaanan Cohney and David Hoffman in their article, "Transactional Scripts in Contract Stacks." They noted that layering the script with natural language could form a 'contract stack,' whereby promises are 'legally-operative.'[963] In effect, Cohney and Hoffman describe the practice of secondary notation, and specifically the act of "commenting out." They argue that in the context of contractual disputes, courts should read code "with its natural language comments and commit logs" as they have "communicative meaning" that should be ascertained and enforced.[964] Fundamentally, they point to the need of 'reading' contracts holistically with code.

[961] "Enactments and Implicature," *AuthoritySpoke* https://authorityspoke.readthedocs.io/en/latest/guides/introduction.html#enactment-objects-and-implication (accessed Jun. 22, 2021).

[962] Consider the use of tense and legal analysis as occasionally "backward-looking." *See* for example "Using Python Template Strings to Represent Legal Explanations," *supra* 959.

[963] *See* Shaanan Cohney and David Hoffman, *Transactional Scripts in Contract Stacks*, 105 MINNESOTA L. REV. 319, 362-363 (2020).

[964] *Id.* at 360.

Their argument becomes particularly significant in light of using secondary notation in an aesthetic manner. That is, legal agreements would be drafted either by (1) interplaying secondary notation with executable code, or (2) writing constraint-based source code that it is both expressive and executable. Again, this has been seen in the examples of vocable code and Harwood's *Class Library*. Rather than stacking, legal codex(t) is a package. It does not compartmentalize between natural language and code but, instead, interlaces them. In this way, code not only performs, but is performative.

Reflecting on aesthetic programming confirms that there may be merit in finding deeper methods of writing code. In continuing to equate code as binary, and as products solely of formal logic, we lose the richness of its expressive potential. More importantly, it maintains the notion that law and computation are incommensurable systems. By experimenting with code as writing, characteristics of code were revealed to be characteristics of natural language. In turn, this demonstrated that the linguistic competence of code has largely been left unexplored. Therefore, the next step is to evaluate the extent to which natural language will continue to be the default tool for legal writing; or whether legal concepts will begin to think through code.

APPENDICES

Appendix B: Series A Term Sheet

TERM SHEET

Company:	[_____], a Delaware corporation.
Securities:	Series A Preferred Stock of the Company ("**Series A**").
Investment Amounts:	$[_] million from [_____] ("**Lead Investor**") $[_] million from other investors Convertible notes and safes ("**Convertibles**") convert on their terms into shadow series of preferred stock (together with the Series A, the "**Preferred Stock**").
Valuation:	$[_] million **post-money** valuation, including an available option pool equal to [__]% of the post-Closing fully-diluted capitalization.
Liquidation Preference:	1x non-participating preference. A sale of all or substantially all of the Company's assets, or a merger (collectively, a "**Company Sale**"), will be treated as a liquidation.
Dividends:	6% noncumulative, payable if and when declared by the Board of Directors.
Conversion to Common Stock:	At holder's option and automatically on (i) IPO or (ii) approval of a majority of Preferred Stock (on an as-converted basis) (the "**Preferred Majority**"). Conversion ratio initially 1-to-1, subject to standard adjustments.
Voting Rights:	Approval of the Preferred Majority required to (i) change rights, preferences or privileges of the Preferred Stock; (ii) change the authorized number of shares; (iii) create securities senior or pari passu to the existing Preferred Stock; (iv) redeem or repurchase any shares (except for purchases at cost upon termination of services or exercises of contractual rights of first refusal); (v) declare or pay any dividend; (vi) change the authorized number of directors; or (vii) liquidate or dissolve, including a Company Sale. Otherwise votes with Common Stock on an as-converted basis.
Drag-Along:	Founders, investors and 1% stockholders required to vote for a Company Sale approved by (i) the Board, (ii) the Preferred Majority and (iii) a majority of Common Stock [(excluding shares of Common Stock issuable or issued upon conversion of the Preferred Stock)] (the "**Common Majority**"), subject to standard exceptions.
Other Rights & Matters:	The Preferred Stock will have standard broad-based weighted average anti-dilution rights, first refusal and co-sale rights over founder stock transfers, registration rights, pro rata rights and information rights. Company counsel drafts documents. Company pays Lead Investor's legal fees, capped at $30,000.

Appendix A

Plain English for Lawyers

FIFTH EDITION

Richard C. Wydick
EMERITUS PROFESSOR OF LAW
UNIVERSITY OF CALIFORNIA, DAVIS

CAROLINA ACADEMIC PRESS
Durham, North Carolina

Contents

Board:	[Lead Investor designates 1 director. Common Majority designates 2 directors.]
Founder and Employee Vesting:	Founders: [_____]. Employees: 4-year monthly vesting with 1-year cliff.
No Shop:	For 30 days, the Company will not solicit, encourage or accept any offers for the acquisition of Company capital stock (other than equity compensation for service providers), or of all or any substantial portion of Company assets.

The “No Shop” is legally binding between the parties. Everything else in this term sheet is non-binding and only intended to be a summary of the proposed terms of this financing.

[COMPANY]

By: _____

Name: _____

Title: _____

Date: _____

[LEAD INVESTOR]

By: _____

Name: _____

Title: _____

Date: _____

BIBLIOGRAPHY

Prolog(ue)

Benjamin Alarie, *The Path of the Law: Towards Legal Singularity*, 66 U. TORONTO L.J. 443(2016)

Kevin Ashley, *Artificial Intelligence and Legal Analytics: New Tools for Law Practice in the Digital Age* (2017).

Joshua Browder, "Law as Code: A Legal System Shaped by Software, *Future* (Jun. 15, 2021) https://future.a16z.com/law-as-code/.

Julie Cohen, *Internet Utopianism and the Practical Inevitability of the Law*, 18 DUKE L. & TECH. REV. 85 (2019).

Mark Fenwick and Erik Vermeulen, "The Lawyer of the Future as 'Transaction Engineer:' Digital Technologies and the Disruption of the Legal Profession," in Marcelo Corrales, Mark Fenwick, and Helena Haapio (eds.) in *Legal Tech, Smart Contracts and Blockchain* (2019).

Mireille Hildebrandt, "Intricate entanglements of law and technology," in *Smart Technologies and the End(s) of Law: Novel Entanglements of Law and Technology* (2015).

Mireille Hildebrandt, "The end of law or Legal Protection by Design," in *Smart Technologies and the End(s) of Law: Novel Entanglements of Law and Technology* (2015).

Mireille Hildebrandt, "'Legal by Design' or 'Legal Protection by Design'" in *Law for Computer Scientists* (2020).

Mireille Hildebrandt, *The adaptive nature of text-driven law*, J. OF CROSS-DISCIPLINARY RESEARCH IN COMPUTATIONAL LAW (CRCL) 1 (2020).

Lawrence Lessig, *Code 2.0* (2nd ed. 2006).

Daniel W. Linna Jr., *The Future of Law and Computational Technologies: Two Sides of the Same Coin*, MIT COMPUTATIONAL LAW REPORT Release 1.0 (2019) available at: https://law.mit.edu/pub/thefutureoflawandcomputationaltechnologies/release/2.

Christopher Markou and Simon F. Deakin, "Is Law Computable? From Rule of Law to Legal Singularity," University of Cambridge Faculty of Law Research Paper (Apr. 30, 2020) available at: https://ssrn.com/abstract=3589184.

Adrienne Mayor, *Gods and Robots: Myths, Machines, and Ancient Dreams of Technology* (2018).

Evgeny Morozov, *To Save Everything, Click Here: The Folly of Technological Solutionism* (2013).

Karen Petroski, *Legal fictions and the limits of legal language*, 9 INT. J. OF L. IN CONTEXT 485 (2013).

Frank Pasquale, *New Laws of Robotics: Defending Human Expertise in the Age of AI* (2020).

Alex "Sandy" Pentland, *A Perspective on Algorithms*, MIT COMPUTATIONAL LAW REPORT Release 1.0 (2019), available at: https://law.mit.edu/pub/aperspectiveonlegalalgorithms/release/3.

Harry Surden, *Artificial Intelligence and Law: An Overview*, 35 GA. ST. U. L. REV. 1305 (2019).

Pierre Schlag, *Commentary: The Aesthetics of American Law*, 115 HARV. L. REV. 1047 (2002)

Noah Waisberg and Dr. Alexander Hudek, *AI for Lawyers* (2021).

Shoshana Zuboff, *The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power* (2019).

The Linguistic Affair

Giorgio Agamben, *The Signature of All Things* (2009).

John L. Austin, *How to Do Things with Words* 16 (2nd ed. 1975).

Francois Cooren, "In the Name of Law: Ventriloquism and Juridical Matters" in Kyle McGee (ed.), *Latour and the Passage of Law* 249 (2015).

Marianne Constable, *Law as Language*, 1 CRITICAL ANALYSIS OF LAW 68 (2014)

Brenda Danet, *Language in the Legal Process,* 14 L. & SOC. REV. 445 (1980)

Jacques Derrida, "Signature Event Context" in *Limited Inc.* (1977).

Stanley Fish, *Is There a Text in This Class? The Authority of Interpretative Communities* (1980).

Stanley Fish, *The Trouble with Principle* (1999).

Lon Fuller, *Legal Fictions*, 25 ILLINOIS L. REV. 363 (1930a).

Michel Foucault, *The Order of Things: An Archaeology of the Human Sciences* (1970).

Peter Goodrich, *Legal Discourse: Studies in Linguistics, Rhetoric and Legal Analysis* (1985).

Clifford Geertz, *Local Knowledge: Further Essays in Interpretative Anthropology* (1983).

Jürgen Habermas, *Between Facts and Norms: Contributions to a Discourse Theory of Law and Democracy* (1996).

Chris Hutton, *Language, Meaning, and the Law* (2009).

Hans Kelsen, *Pure Theory of Law* (first published in 1934, Max Knight trans., 1967).

Duncan Kennedy, *A Semiotics of Legal Argument*, 3 Collected Courses of the Academy of European Law 317, 351 (1994).

Niklas Luhmann, *Law as a Social System* 146 (2004).

David Mellinkoff, *The Language of Law* (1963).

George Orwell, "Politics and the English Language," in *Why I Write* (2004)

Richard A. Posner, *Law and Literature: A Misunderstood Relation* (1988).

Peter M. Tiersma, *Legal Language* (1999), available at: http://languageandlaw.org/LEGALLANG/LEGALLANG.HTM.

Geoffrey Samuel, *Is legal reasoning like medical reasoning?*, 35 LEGAL STUDIES 323 (2014).

Ferdinand de Saussure, *Course in General Linguistics* (Bloomsbury Revelations ed. 2013).

John Searle, *A Classification of Illocutionary Acts*, 5 Language in Society 1 (1976).

James Boyd White, *The Legal Imagination* (1973).

Ludwig Wittgenstein, *Philosophical Investigations* (2nd ed. 1958).

Language Lego

Barbara Abbott, *Presuppositions and common ground*, 31 LINGUISTICS AND PHILOSOPHY 523 (2008).

Betty J. Birner, *Language and Meaning* (2018).

Andrew Carnie, *Syntax: A Generative Introduction* (3rd ed. 2013).

Paul Elbourne, *Meaning: A slim guide to semantics* (2011).

Michael Genesereth and Vinay K. Chaudhri, *Introduction to Logic Programming* (2020).

H.P. Grice, "Logic and Conversation" in Cole et al. (eds.), *Syntax and semantics 3: Speech Arts* (1975).

Christopher Potts, *The Logic of Conventional Implicatures* (2005).

White City v. PR Restaurants, No. 2006196313 (Mass. Cmmw. Oct. 31, 2006).

Michael J. Reddy, "A case of frame conflict in our language" in A. Ortony (ed.) Metaphor and Thought (2nd ed. 1993).

Michael L. Scott, *Programming Language Pragmatics* (4th ed. 2016).

Benjamin Lee Whorf, *Language, Thought, and Reality* (1956).

Case Studies on Translation

Layman E. Allen, *Symbolic Logic: A Razor-Edged Tool for Drafting and Interpreting Legal Documents*, 66 YALE L. J. 833 (1957)

Layman E. Allen, "Language, Law, and Logic: Plain Legal Drafting for the Electronic Age," B. Niblett (ed.) *Computer Science and Law* 76 (1980).

Giosuè Baggio, *Meaning in the Brain* 62 (2018).

Bailey v. United States, 516 U.S. 137 (1995).

George Boole, *The Laws of Thought* (1854).

Anthony J. Casey & Anthony Niblett, *The Death of Rules and Standards*, Coase-Sandor Working Paper Series in Law and Economics No. 738 (2015).

Anthony J. Casey and Anthony Niblett, *Self-Driving Contracts*, 43 J. OF CORP. LAW. 101 (2017).

Rudolf Carnap, *Logical Syntax of Language* (Routledge English ed. reprint, 2014).

Ilias Chalkidis and Dimitrios Kampas, *Deep Learning in law: early adaptation and legal word embeddings trained on large corpora*, 27 ARTIFICIAL INTELLIGENCE AND LAW 171 (2018).

Noam Chomsky, "Remarks on Nominalization," in R.A. Jacobs and P.S. Rosenbaum (eds.), *Readings in English Transformational Grammar* (1970).

Walter Daelemans and Koenraad De Smelt, *Default Inheritance in an Object-Oriented Representation of Linguistic Categories*, 41 INT'L J. OF HUMAN COMPUTER STUDIES 149 (1994).

Keith Devlin, *Goodbye Descartes: The End of Logic and The Search for a New Cosmology of the Mind* (1997).

Henning Diedrich, *Lexon: Digital Contracts* (2020).

Phong-Khac Do et al., *Legal Question Answering using Ranking SVM and Deep Convolutional Neural Network*, TENTH INTERNATIONAL WORKSHOP ON JURIS-INFORMATICS (2017), available at: https://arxiv.org/abs/1703.05320.

Ron Dolin, "XML in Law: An Example of the Role of Standards in Legal Informatics" (forthcoming 2021).

Zev J. Eigen, *Empirical Studies of Contract*, Faculty Working Paper 204 (2012), available at: https://scholarlycommons.law.northwestern.edu/cgi/viewcontent.cgi?article=1203&context=facultyworkingpapers.

Zev J. Eigen, *When and Why Individuals Obey Contracts: Experimental Evidence of Consent, Compliance, Promise, and Performance*, 41 J. OF LEGAL STUDIES 67 (2012).

David Freeman Engstrom and Daniel E. Ho, "Artificially Intelligent Government: A Review and Agenda" in Roland Vogl (ed.), *Big Data Law* (2020).

John Rupert Firth, *The Technique of Semantics*, 34 TRANS. PHILOS. SOC. 36 (1935).

Jerry Fodor and Ernest Lepore, *The red herring and the pet fish: Why concepts still can't be prototypes*, 58 COGNITION 253 (1996).

Yulia Frumer, *Translating Worlds, Building Worlds: Meteorology in Japanese, Dutch, and Chinese*, 109 ISIS 326 (2018).

Katrin Fundel, Robert Küffner, and Ralf Zimmer, *RelEx – Relation extraction using dependency parse trees*, 23 BIOINFORMATICS 365 (2006).

Michael Genesereth, "The Legacy of Hammurabi" (Mar. 17, 2021), available at: https://law.stanford.edu/2021/03/17/the-legacy-of-hammurabi/.

Joseph A. Grundfest and A.C. Pritchard, *Statutes with Multiple Personality Disorders: The Value of Ambiguity in Statutory Design and Interpretation*, 54 STAN. L. REV. 627 (2002).

H.L.A. Hart, *The Concept of Law* 1961).

Mireille Hildebrandt, "Law *as* computation in the era of artificial intelligence: Speaking law to the power of statistics," Draft for SPECIAL ISSUE U. TORONTO L.J., 13 (2019).

Douglas Hofstadter, *Gödel, Escher, Bach* preface-3 (Twentieth-anniversary ed. 1999).

Douglas Hofstadter, *The Shallowness of Google Translate*, The Atlantic (January 30, 2018), https://www.theatlantic.com/technology/archive/2018/01/the-shallowness-of-google-translate/551570/.

Oliver Wendell Holmes Jr., *The Path of Law*, 10 HARV. L. REV. 457 (1897).

Oliver Wendell Holmes Jr., *The Common Law* Lecture I: Early Forms of Liability (Project Gutenberg eBook, 2000), available at: https://www.gutenberg.org/files/2449/2449-h/2449-h.htm#link2H_4_0001.

Sheila Jasanoff, *Can Science Make Sense of Life?* (2019).

Michael Jeffrey, *What Would an Integrated Development Environment for Law look like*?, MIT COMPUTATIONAL LAW REPORT Release 1.1 (2020), available at: https://law.mit.edu/pub/whatwouldanintegrateddevelopmentenvironmentforlawlooklike.

Daniel Martin Katz et. al., *Complex societies and the growth of the law*, Sci Rep 10, 18737 (2020), available at: https://doi.org/10.1038/s41598-020-73623-x.

Duncan Kennedy, *Legal Reasoning: Collected Essays* (Davies Group Publishers, 2008).

Katja Langenbucher, *Economic Transplants: On Lawmaking for Corporations and Capital Markets* 8-9 (2017).

Lionel A. Levert, "Harmonization and Dissonance: Language and Law in Canada and Europe," Department of Justice Canada, *Bijuralism and Harmonization: Genesis* (May 7, 1999) https://www.justice.gc.ca/eng/rp-pr/csj-sjc/harmonization/hfl-hlf/b1-f1/bf1e.html.

Kingsley Martin, "Legal Technology Barriers – Understanding Language and Exercising Judgment," Legal Executive Institute (September 24, 2015), https://www.legalexecutiveinstitute.com/legal-technology-barriers-understanding-language-and-exercising-judgement/.

Christopher Markou and Simon Deakin, *Ex Machina Lex: The Limits of Legal Computability*, Working Paper (2019), available at SSRN: https://ssrn.com/abstract=3407856.

Denis Merigoux and Liane Huttner, *Catala: Moving Towards the Future of Legal Expert Systems*, HAL ARCHIVES-OUVERTES (2020).

Muscarello v. United States, 524 U.S. 125 (1998).

New Zealand Law Foundation Law and Information Policy Project, *Legislation as Code for New Zealand: Opportunities, Risks, and Recommendations* 3 (2021).

OECD Observatory of Public Sector Innovation, *Cracking the Code: Rulemaking for Humans and Machines* (2020).

Monica Palmirani and Fabio Vitali, "Akoma-Ntoso for Legal Documents," Giovanni Sartor et. al (eds.), *Legislative XML for the Semantic Web* (2011).

Frank Pasquale, *A Rule of Persons, Not Machines: The Limits of Legal Automation*, 87 GEO. WASH. L. REV. 2 (2019)

Frank Pasquale, *The Substance of Poetic Procedure: Law & Humanity in the Work of Lawrence Joseph*, 32 LAW & LITERATURE 1 (2020).

Katherina Pistor and Chenggang Xu, *Incomplete* Law, 35 NYU J. INT'L L. & POL. 931 (2003).

Eric Posner and Adrien Vermeule*, Inside or Outside the System?*, 80 U. CHI. L. REV. 1743 (2013).

Richard A. Posner, *The Incoherence of Antonin Scalia*, New Republic (August 24, 2012), http://www.newrepublic.com/node/106441/print.

Richard A. Posner, *The Law and Economics of Contract Interpretation*, 83 TEXAS L. REV. 1581(2005).

Gerald J. Postema, *Implicit Law*,13 LAW AND PHILOSOPHY 361 (1994).

Joseph Raz, *Legal Principles and the Limits of Law*, 81 YALE L. J. 823 (1972).

Richard M. Re and Alicia Solow-Niederman, *Developing Artificially Intelligent Justice*, 22 STAN. TECH. L. REV. 242 (2019).

Neil M. Richards and William D. Smart, "How should the law think about robots?" in Ryan Calo et al, eds, *Robot Law* (2018).

Eleanor Rosch and Carolyn B. Mervis, *Family resemblances: Studies in the internal structure of categorie*s, 7 COGNITIVE PSYCHOLOGY 573 (1975).

Geoffrey Samuel, *The Reality of Contract in English Law*, 13 TULSA L. J. 508, 523 (2013).

Antonin Scalia and Bryan A. Garner, *Reading Law: The Interpretation of Legal Texts* xxvii-xxix (2012).

Frederick Schauer, "Ruleness," Dupret Baudouin et al. (eds.) *Legal Rules in Practice* (2021 Forthcoming).

Smith v. United States, 508 U.S. 223 (1993).

Henry E. Smith, *Modularity in Contracts: Boilerplate and Information Flow*, 10 MICH. L. REV. 1175 (2006).

Fabio Vitali, "A Standard-Based Approach for the Management of Legislative Documents," Giovanni Sartor et. al (eds), *Legislative XML for the Semantic Web* (2011).

Langdon Winner, *Do Artifacts Have Politics?*, 109 DAEDALUS 121 (1980).

Meng Weng Wong, *Rules as Code – Seven Levels of Digitisation*, RESEARCH COLLECTION SCHOOL OF LAW (2020).

Michael J.B. Wood, *Drafting Bilingual Legislation in Canada: Examples of Beneficial Cross-Pollination between Two Language Versions*, 17 STATUTE. L. REV 66 (1996).

Stephen Wolfram, "Computational Law, Symbolic Discourse, and the AI Constitution," in Ed Walters (ed.), *Data-Driven Law: Data Analytics and New Legal Services* (2019).

Richard C. Wydick, *Plain English for Lawyers* (2005).

Weaving the Code

Mikhail Bakhtin, *Dialogic Imagination: Four Essays* (1981).

Emily M. Bender and Alexander Koller, "Climbing Towards NLU: On Meaning, Form, and Understanding in the Age of Data," *Proceedings of the 58th Annual Meeting of the Association of Computational Linguistics* (July 2020) available at: https://aclanthology.org/2020.acl-main.463/.

Alexander Campolo, "*Thinking, Judging, Noticing, Feeling": John W. Tukey against the Mechanization of Inferential Knowledg*e, 5 KNOW: A JOURNAL ON THE FORMATION OF KNOWLEDGE 83 (2021).

Geoffrey Cox, Alex McLean, and Adrian Ward, "The Aesthetics of Generative Code," *International Conference on Generative Art* (2000).

Laurence Diver, *Computational legalism and the affordance of delay in law*, J. OF CROSS-DISCIPLINARY RESEARCH IN COMPUTATIONAL LAW [CRCL] 6 (December 2020).

Sandra Fredman, *Intersectional Discrimination in EU Gender Equality and Non-Discrimination Law* 31 (2016), available at http://ohrh.law.ox.ac.uk/wordpress/wpcontent/up.

Mireille Hildebrandt, "Code Driven Law Scaling the Past and Freezing the Future," Christopher Markou and Simon Deakin (eds.) in *Critical Perspectives in Law and Artificial Intelligence* (2020).

Jerry Hobbs et al., *Interpretation as Abduction*, 63 ARTIFICIAL INTELLIGENCE 69 (1993).

Jerry R. Hobbs et. al, "The TACITUS System," in *Robust Processing of Real-World Natural-Language Texts*, https://www.isi.edu/~hobbs/robust/node2.html (Feb. 24, 2004).

Erik J. Larson, *The Myth of Artificial Intelligence: Why Computers Can't Think the Way We Do* (2021).

Bruno Latour, "Where are the Missing Masses? The Sociology of a Few Mundane Artifacts," in Bijker and Law (eds.), *Shaping Technology/Building Society: Studies in Sociotechnical Change* (1992).

Jeffrey M. Lipshaw, *The Persistence of "Dumb" Contracts*, 2 STAN. J. BLOCKCHAIN L. & POL'Y 1 (2019), available at: https://stanford-jblp.pubpub.org/pub/persistence-dumb-contracts/release/1.

Lin Ma and Jaap van Brakel, *Fundamentals of Comparative and Intercultural Philosophy* (2016).

Mark C. Marino, *Critical Code Studies* (2020)

George Pavlakos, "Two Concepts of Objectivity," in George Pavlakos (ed.), *Law, Rights, and Discourse: The Legal Philosophy of Robert Alexy* (2007).

Loss Pequeño Glazier, "Code as Language," *Leonardo Electronic Almanac* (2006)

Paul Ricoeur, *From Text to Action* (1991).

Dan Sperber and Deirdre Wilson, *Relevance: Communication and Cognition* (1986).

Epilog(ue)

Shaanan Cohney and David Hoffman, *Transactional Scripts in Contract Stacks*, 105 MINNESOTA L. REV. 319 (2020).

Geoffrey Cox and Alex McLean, *Speaking Code: Coding as Aesthetic and Political Expression* (2012).

Geoffrey Cox, Alex McLean, and Adrian Ward, "Coding Praxis: Reconsidering the Aesthetics of Code," in Olga Goriunova and Alexei Shulgin (eds.), *read_me, Software Art and Cultures* (2004).

Cecile Crutzen and Erna Kotkamp, "Object Orientation" in Matthew Fuller (ed.) *Software Studies* (2008).

Eva Heisler, "Winnie Soon, Time, Code, and Poetry," *Asymptote Journal* (Jan. 2020) https://www.asymptotejournal.com/visual/winnie-soon-time-code-and-poetry/.

Bruno Latour, "On actor-network theory. A few clarifications plus more than a few complications," available at: http://www.bruno-latour.fr/sites/default/files/P-67%20ACTOR-NETWORK.pdf.

Winnie Soon and Geoff Cox, *Aesthetic Programming: A Handbook of Software Studies* (2020)

Linda Xie, "Composability is Innovation," *Future* (Jun. 15, 2021) https://future.a16z.com/how-composability-unlocks-crypto-and-everything-else/.

Resumés de la Thèse

NOM: Ma

Prenom: Megan

L'intitulé de la these: Story of a Legal Codex(t): Writing Law in Code

Nom de votre directrice de thèse: MUIR WATT, Horatia

Resumè en anglais:

How is the law measured? For long, it appeared that the law cannot be measured. While there are standards and processes, the law was not regarded as quantifiable. Only in the advent of recent technological advancements in law have there been considerations for metrics. These technologies sought to tackle the legal field's inherent protectionism fueled by deep asymmetries in information. Consequently, the rise in legal 'metrics' stems from an access to justice perspective. The assumption is that in making the law more quantifiable, knowledge that has been historically opaque and inaccessible outside of the legal community may be revealed.

Alternatively, it may be argued that the law has always been measurable. Words, through linguistic devices, have shaped legal meaning. In effect, the law conceivably has been measured by its words. In fact, "law *exists as* text" (Hildebrandt, 2015). I further this line of thinking by investigating natural language as the key vessel through which the law has manifested itself. Does the law depend on natural language to do its work? Importantly, is the language sufficient at housing legal norms?

This dissertation seeks to tell a narrative. Broadly, it chronicles the story of law's intimate relationship with language. But more specifically, the thesis details the law's recent encounter with the digital. When law met technology, its relationship with language changed, invoking skepticism around its fitness for the conveyance of legal concepts. With the introduction of an innovative player – code – the law had perceivably found its new linguistic match. As a result, code was tested for its ability to perform and accommodate for the law's demands. Ultimately, confronted by natural language and code, the law is asked whether code can be its language.

Resumé en français

Comment mesure-t-on le droit ? Longtemps, le droit semblait résister à la mesure. Bien qu'il existe des normes et des processus, le droit n'était pas considéré comme quantifiable. Ce n'est qu'avec l'avènement des récentes avancées technologiques dans le domaine du droit que l'on a commencé à envisager une telle quantification. Ces technologies ont cherché à s'attaquer au protectionnisme inhérent au domaine juridique, alimenté par de profondes asymétries d'information. Par conséquent, l'essor de la "métrique" juridique découle d'une perspective d'accès à la justice. L'hypothèse est qu'en rendant le droit plus quantifiable, des connaissances historiquement opaques et inaccessibles en dehors de la communauté juridique peuvent être révélées.

On peut également faire valoir que le droit a toujours été mesurable. Les mots, par le biais de dispositifs linguistiques, ont façonné la signification juridique. En effet, il est concevable que le droit ait été mesuré par ses mots. En effet, "le droit existe en tant que texte" (Hildebrandt, 2015).

J'approfondis cette ligne de pensée en examinant le langage naturel en tant que vecteur clé à travers lequel le droit s'est manifesté. La loi dépend-elle du langage naturel pour faire son travail ? Plus important encore, le langage est-il suffisant pour abriter les normes juridiques ?

Cette thèse cherche à raconter une histoire. De manière générale, elle relate l'histoire de la relation intime du droit avec le langage. Mais plus spécifiquement, la thèse détaille la rencontre récente du droit avec le numérique. Lorsque le droit a rencontré la technologie, sa relation avec le langage a changé, suscitant le scepticisme quant à son aptitude à transmettre des concepts juridiques. Avec l'introduction d'un acteur innovant - le code - le droit a visiblement trouvé sa nouvelle adéquation linguistique. En conséquence, le code a été mis à l'épreuve quant à sa capacité à fonctionner et à répondre aux exigences du droit. Finalement, confronté au langage naturel et au code, le droit se demande si le code peut être son langage.